From the Textual to the Technological: Documents and Structure in a Digital Age
USC 020059 Lecture Notes

Giselda Beaudin

International Programs
Rollins Colege, Winter Park, FL, USA
gbeaudin@rollins.edu

Michael Kohlhase

School of Engineering & Science
Jacobs University, Bremen Germany
m.kohlhase@jacobs-university.de

January 21, 2014

Preface

This Document

This document contains the course notes for the University Study Course 020059 held at Jacobs University Bremen. So far this course has had two installments:

- "Text and Digital Media" held in the spring semester 2011 by Profs. Michael Kohlhase and Thomas Rommel.
- "From the Textual to the Technological: Documents and Structure in a Digital Age" held in the Intersession 2014 by Giselda Beaudin (Rollins College) and Prof. Michael Kohlhase.

The CS part is more or less the same (apart from improvements), and the literary part has been contributed by Giselda Beaudin for the integrated course notes in 2014.

Contents: The document mixes the slides presented in class with comments of the instructor to give students a more complete background reference.

Caveat: This document is made available for the students of this course only. It is still a draft and will develop over the course of the current course and in coming academic years.

Licensing: This document is licensed under a Creative Commons license that requires attribution, allows commercial use, and allows derivative works as long as these are licensed under the same license.

Knowledge Representation Experiment: This document is also an experiment in knowledge representation. Under the hood, it uses the sTEX package [Koh08, Koh13], a TEX/LATEX extension for semantic markup, which allows to export the contents into the eLearning platform PantaRhei.

Comments and extensions are always welcome, please send them to the author.

EdN:1 Other Resources: [1] [2]

EdN:2 Comments: Comments and extensions are always welcome, please send them to the author.

Course Concept

Aims: The University Study Course 020059 is a one-semester course taught to students of all majors at Jacobs University. The concept of a University Study Course (USC) is somewhat peculiar to Jacobs University, USCs aim to give students a trans-disciplinary look at a particular topic; here documents as technical artefacts and as communication objects.

EdN:3 Prerequisites: [3] As a consequence, the USC course does not make any assumptions about prior knowledge, and introduces all the necessary material, developing it from first principles. To compensate for this, the course progresses very rapidly and leaves much of the actual learning experience to homework problems and student-run tutorials.

Course Contents

EdN:4 [4]

Acknowledgments

EdN:5 Materials: [5]

GenCS Students: The following students have submitted corrections and suggestions to this and earlier versions of the notes: Anca Dumitrache, Calin Lupitu, Bogdan Matican, Isabel Schlie.

[1] EdNote: describe the discussions in Panta Rhei
[2] EdNote: Say something about the problems
[3] EdNote: MK: say something and adapt the following
[4] EdNote: MK: Give an overview, once we have it.
[5] EdNote: MK: only course notes, but see also ...

Syllabus

The course will consist of 6 75-minute slots per day, which can be lectures (usually 3-4) or supervised labs (the rest: 2-3).

13 January		
8:15-9:30	Introduction	Quiz 1 and Course and Review of Syllabus
9:45-11:00	Lecture 1	
11:15-12:30	Lecture 2	Pre-course readings and our experience/relationship with technology
14:15-15:30	Lecture 3	
15:45-17:00	Lecture 4	The Structures of Language
17:15-18:30	Lab 1	assign groups
14 January		
8:15-9:30	Lecture 5	Quiz 2 and Deconstruction
9:45-11:00	Lab 2	
11:15-12:30	Lecture 6	
14:15-15:30	Lab 3	
15:45-17:00	Lecture 7	The Structure(s) of Digital Texts
17:15-18:30	Lab 4	Webpage Analysis using Structuralism and Deconstruction
15 January		
8:15-9:30	Lecture 8	Quiz 3 and
9:45-11:00	Lab 5	
11:15-12:30	Lecture 9	Privilege, language and the Digital
14:15-15:30	Lab 6	Poem in code
15:45-17:00	Lecture 10	
17:15-18:30	Lecture 11	Welcome to the Desert of the Real
16 January		
8:15-9:30	Lecture 12	Quiz 4 and
9:45-11:00	Lab 7	
11:15-12:30	Lecture 13	A Journey Through the Hyperreal
14:15-15:30	Lab 8	Identifying Examples of the Hyperreal
15:45-17:00	Lecture 14	
17:15-18:30	Lab 9	
17 January		
8:15-9:30	Lecture 15	Quiz 5 and I'm So Meta Even This Acronym
9:45-11	Lecture 16	
11:15-12:30	Lecture 17	Privacy, Performance, and Identity
14:15-15:30	Lecture 18	
15:45-17:00	Lab 10	Weekend Project
17:15-18:30	Lab 11	Weekend Project
20 January		
8:15-9:30	Lecture 19	Quiz 6 and The Digital Generation?
9:45-11:00	Lab 12	Blog analysis
11:15-12:30	Lecture 20	
14:15-15:30	Lecture 21	Digital Immortality
15:45-17:00	Lecture 22	
17:15-18:30	Lab 13	
21 January		
8:15-9:30	Lecture 23	Quiz 7 and
9:45-11:00	Lecture 24	The Zombie Apocalypse
11:15-12:30	Lab 14	Presentations of Weekend Projects
14:15-15:30	Lab 15	Presentations of Weekend Projects

15:45-17:00	Conclusion	Final Discussion

Contents

Chapter 1

Administrativa

We will now go through the ground rules for the course. This is a kind of a social contract between the instructors and the students. Both have to keep their side of the deal to make the acquaintance with issues about "text and digital media" as efficient and painless as possible.

1.1 Grades

Now we come to a topic that is always interesting to the students: the grading scheme.

Prerequisites, Requirements, Grades

▷ Prerequisites: Motivation, Interest, Curiosity, hard work

▷ you can do this course if you want!

▷ Grades:

Lab Work/Homework	30%
Quizz(es)	30%
Weekend Project	30%
Attendance and Wakefulness	10%

▷ TDM Teams: Homeworks will be solved and submitted in teams of three (one from CS, one from SHSS), which will be formed for the course in the beginning.

▷ Rationale: We want to have knowledge transfer (between the disciplines.)

1.2 Homeworks, Submission, and Cheating

Homework assignments

▷ Goal: Reinforce and apply what is taught/discussed in class.

▷ homeworks: will be practical writing assgignmenst in a variety of geners and formats (take time to solve)

▷ admin: To keep things running smoothly

- ▷ Homeworks will be posted on PantaRhei
- ▷ Homeworks are handed in electronically in `JGrader`(plain text, Postscript, PDF, . .)
- ▷ discuss problems on PantaRhei (Profs/TAs/students can help you!)

▷ Homework discipline:

- ▷ start early! (many assignments need more than one evening's work)
- ▷ Don't start by sitting at a blank screen
- ▷ Humans will be trying to understand the text/code/math when grading it.

Homework assignments are a central part of the course, they allow you to review the concepts covered in class, and practice using them.

Homework Submissions, Grading, Tutorials

▷ Submissions: We use Heinrich Stamerjohanns' `JGrader` system

- ▷ submit all homework assignments electronically to `https://jgrader.de`.
- ▷ you can login with your Jacobs account and password.(should have one!)
- ▷ feedback/grades to your submissions
- ▷ get an overview over how you are doing! (do not leave to midterm)

▷ Tutorials: select a tutorial group and actually go to it regularly

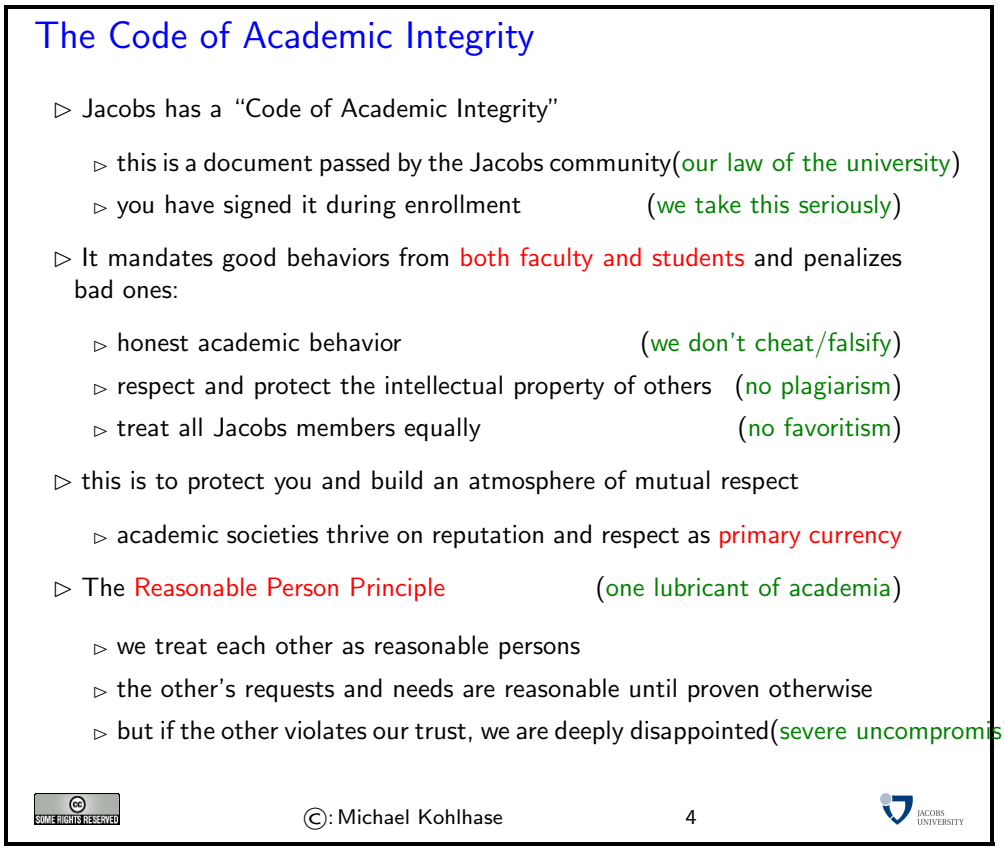
- ▷ to discuss the course topics after class(lectures need pre/postparation)
- ▷ to discuss your homework after submission(to see what was the problem)
- ▷ to find a study group(probably the most determining factor of success)

The next topic is very important, you should take this very seriously, even if you think that this is just a self-serving regulation made by the faculty.

All societies have their rules, written and unwritten ones, which serve as a social contract among its members, protect their interestes, and optimize the functioning of the society as a whole. This is also true for the community of scientists worldwide. This society is special, since it balances intense cooperation on joint issues with fierce competition. Most of the rules are largely unwritten; you are expected to follow them anyway. The code of academic integrity at Jacobs is an attempt to put some of the aspects into writing.

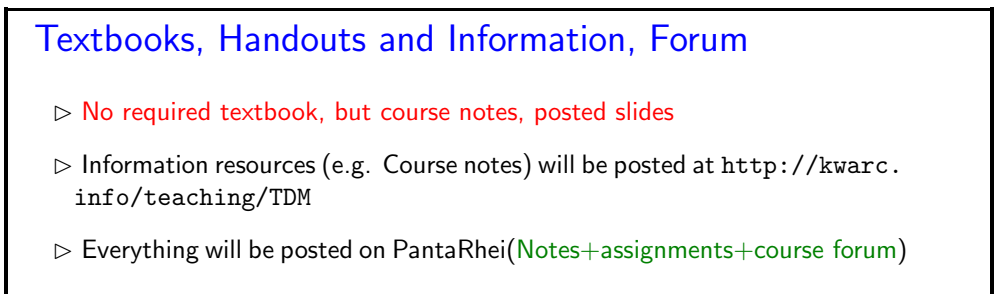
It is an essential part of your academic education that you learn to behave like academics, i.e. to function as a member of the academic community. Even if you do not want to become a scientist in the end, you should be aware that many of the people you are dealing with have gone through an academic education and expect that you (as a graduate of Jacobs) will behave

by these rules.



To understand the rules of academic societies it is central to realize that these communities are driven by economic considerations of their members. However, in academic societies, the primary good that is produced and consumed consists in ideas and knowledge, and the primary currency involved is academic reputation[1]. Even though academic societies may seem as altruistic — scientists share their knowledge freely, even investing time to help their peers understand the concepts more deeply — it is useful to realize that this behavior is just one half of an economic transaction. By publishing their ideas and results, scientists sell their goods for reputation. Of course, this can only work if ideas and facts are attributed to their original creators (who gain reputation by being cited). You will see that scientists can become quite fierce and downright nasty when confronted with behavior that does not respect other's intellectual property.

1.3 Resources



[1] Of course, this is a very simplistic attempt to explain academic societies, and there are many other factors at work there. For instance, it is possible to convert reputation into money: if you are a famous scientist, you may get a well-paying job at a good university,...

- announcements, contact information, course schedule and calendar
- discussion among your fellow students(careful, we will occasionally check for academic integrity!)
- `http://panta.kwarc.info` (follow instructions there)
- if there are problems send e-mail to `m.fieraru@jacobs-university.de`

No Textbook: Due to the special circumstances discussed above, there is no single textbook that covers the course. Instead we have a comprehensive set of course notes (this document). They are provided in two forms: as a large `PDF` that is posted at the course web page and on the PantaRhei system. The latter is actually the preferred method of interaction with the course materials, since it allows to discuss the material in place, to play with notations, to give feedback, etc. The `PDF` file is for printing and as a fallback, if the PantaRhei system, which is still under development develops problems.

Software/Hardware tools

- You will need computer access for this course(come see me if you do not have a computer of your own)
- we recommend the use of standard software tools
  - the `emacs` and `vi` text editor (powerful, flexible, available, free)
  - `UNIX` (`linux`, `MacOSX`, `cygwin`) (prevalent in CS)
  - `FireFox` (just a better browser (for Math))
- learn how to touch-type NOW (reap the benefits earlier, not later)

Touch-typing: You should not underestimate the amount of time you will spend typing during your studies. Even if you consider yourself fluent in two-finger typing, touch-typing will give you a factor two in speed. This ability will save you at least half an hour per day, once you master it. Which can make a crucial difference in your success.

Touch-typing is very easy to learn, if you practice about an hour a day for a week, you will re-gain your two-finger speed and from then on start saving time. There are various free typing tutors on the network. At `http://typingsoft.com/all_typing_tutors.htm` you can find about programs, most for windows, some for linux. I would probably try `Ktouch` or `TuxType`

Darko Pesikan (one of the previous TAs) recommends the `TypingMaster` program. You can download a demo version from `http://www.typingmaster.com/index.asp?go=tutordemo`

You can find more information by googling something like "learn to touch-type". (goto `http://www.google.com` and type these search terms).

Next we come to a special project that is going on in parallel to teaching the course. I am using the coures materials as a research object as well. This gives you an additional resource, but may affect the shape of the coures materials (which now server double purpose). Of course I can use all the help on the research project I can get, so please give me feedback, report errors and shortcomings, and suggest improvements.

Experiment: E-Learning with OMDoc/PantaRhei

- My research area: deep representation formats for (mathematical) knowl-

edge

- ▷ Application: E-learning systems (represent knowledge to transport it)
- ▷ Experiment: Start with this course (Drink my own medicine)
  - ▷ Re-Represent the slide materials in OMDoc (Open Math Documents)
  - ▷ Feed it into the PantaRhei system (`http://panta.kwarc.info`)
  - ▷ Try it on you all (to get feedback from you)
- ▷ Tasks (Unfortunately, I cannot pay you for this; maybe later)
  - ▷ help me complete the material on the slides(what is missing/would help?)
  - ▷ I need to remember "what I say", examples on the board. (take notes)
- ▷ Benefits for you (so why should you help?)
  - ▷ you will be mentioned in the acknowledgements (for all that is worth)
  - ▷ you will help build better course materials(think of next-year's students)

Part I

Tools and Concepts

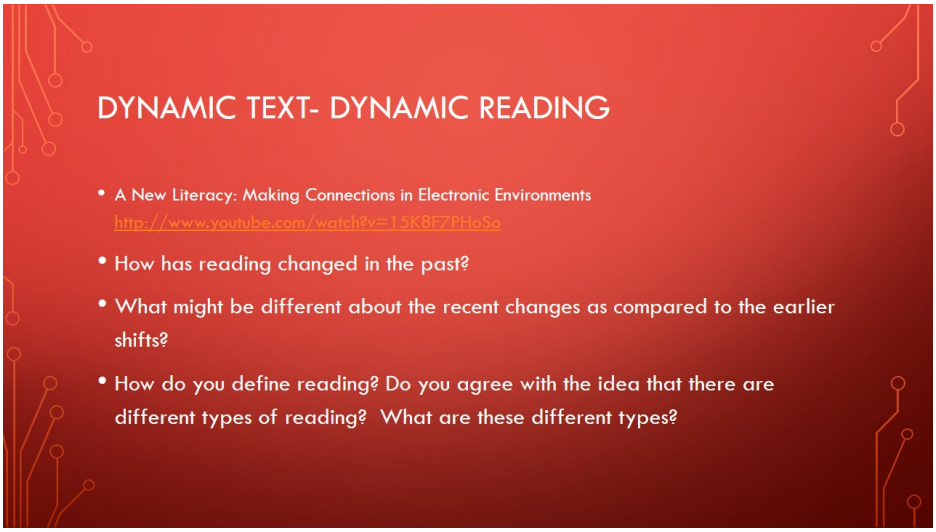
In this part of the Course we will introduce and discuss the main conceptual and technological tools use in the course[6] EdN:6

[6] EdNote: MK: continue

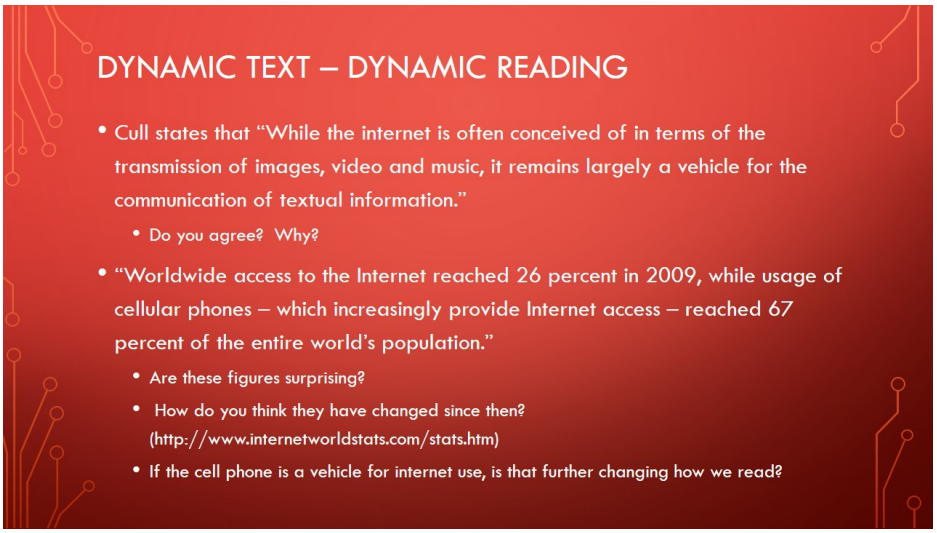
Chapter 2

Reading and our Experience/Relationship with Technology

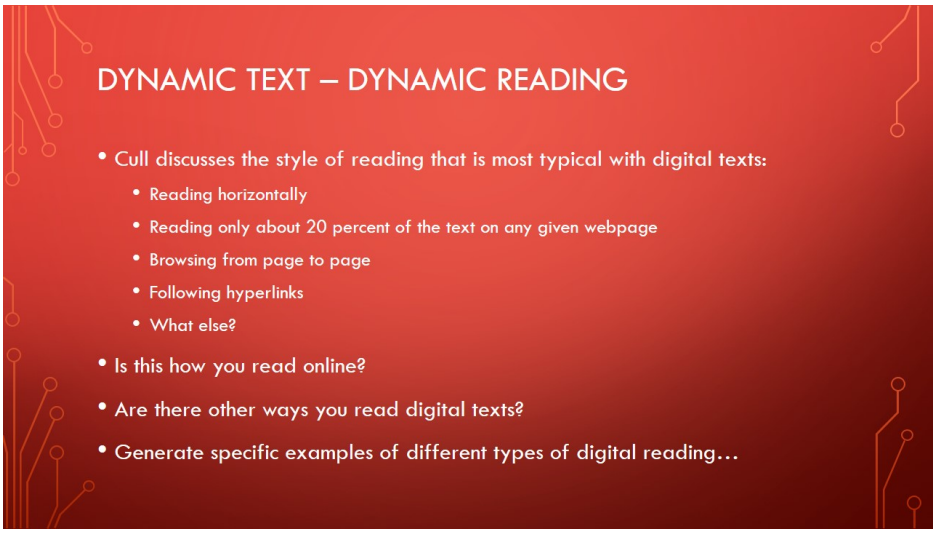
2.1 Reading in the Internet Era



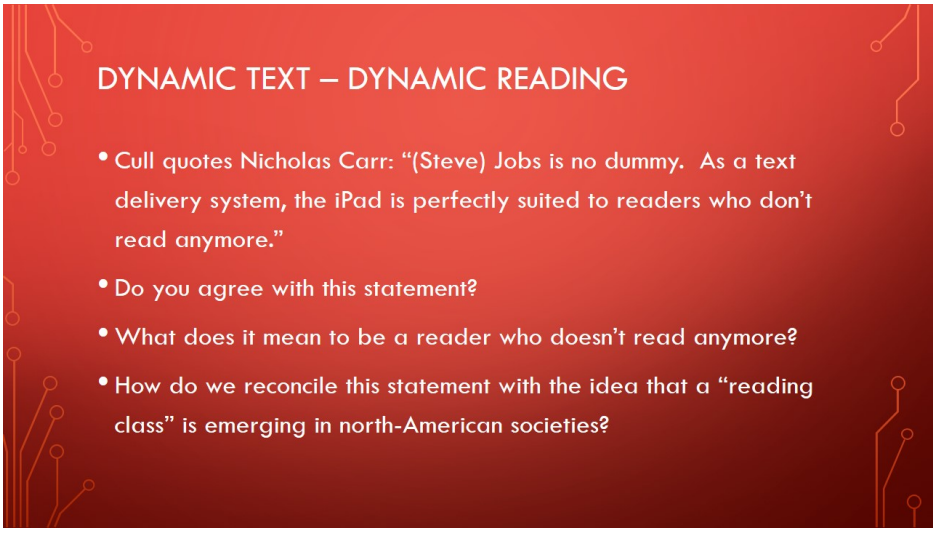
Slide 8



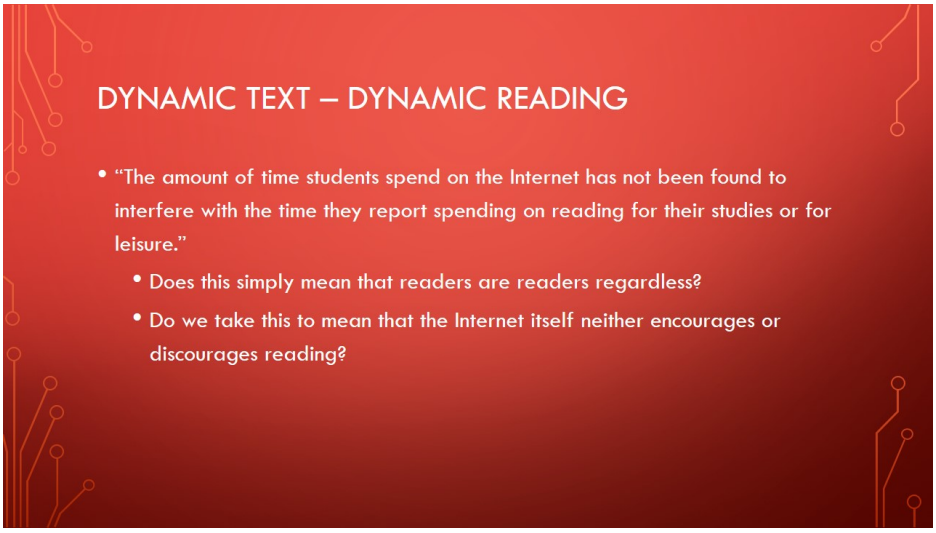
Slide 9



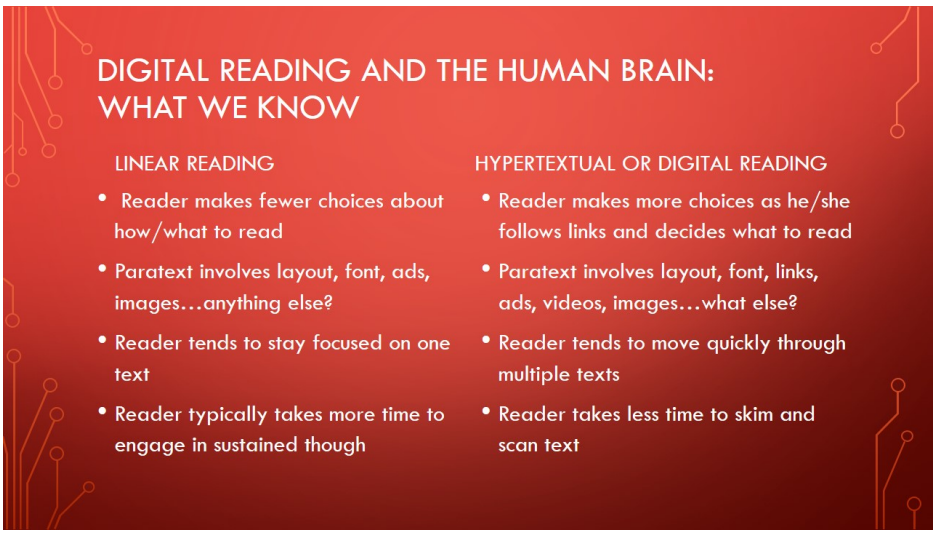
Slide 10



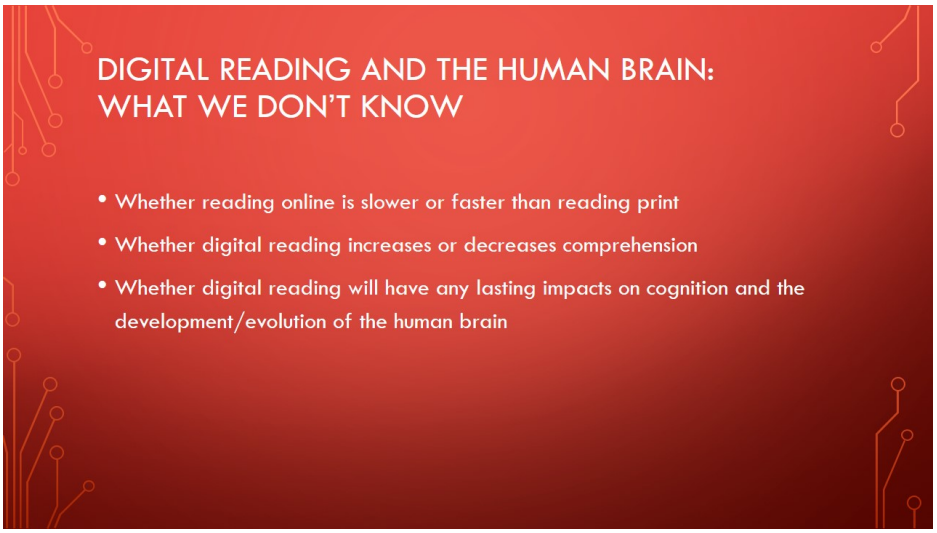
Slide 11



Slide 12

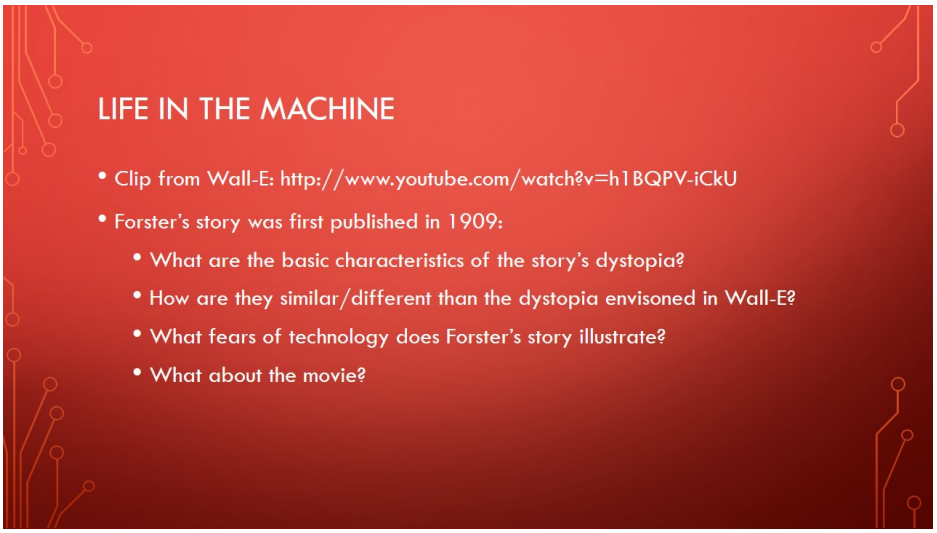


Slide 13

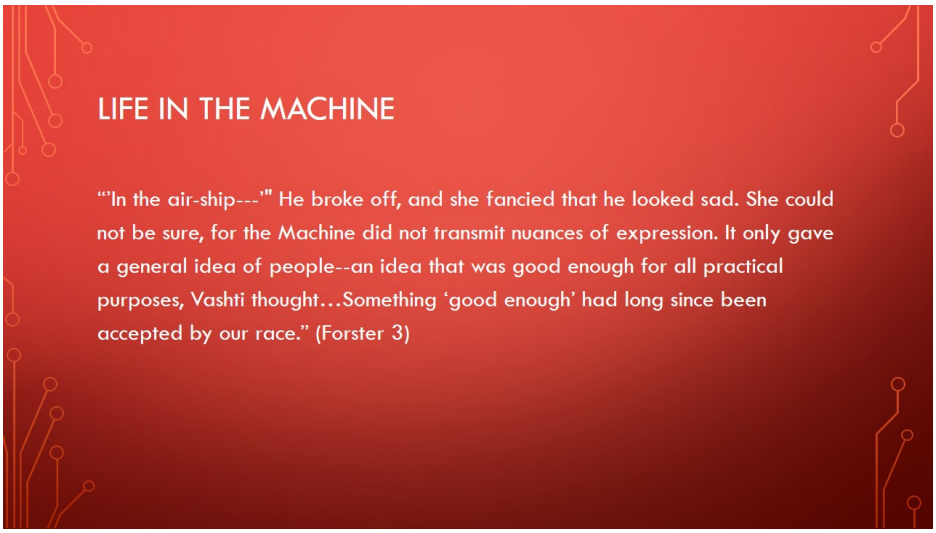


Slide 14

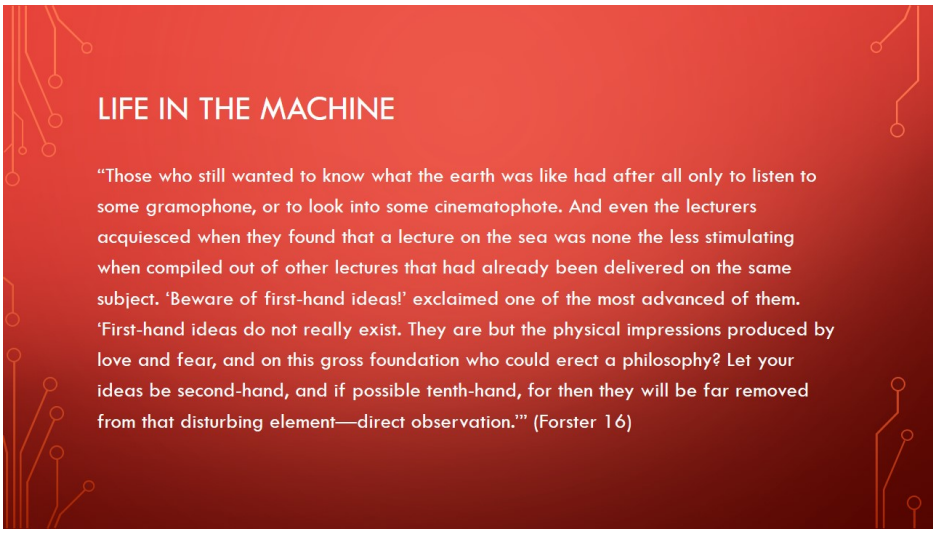
2.2 Our Relationship with Technology



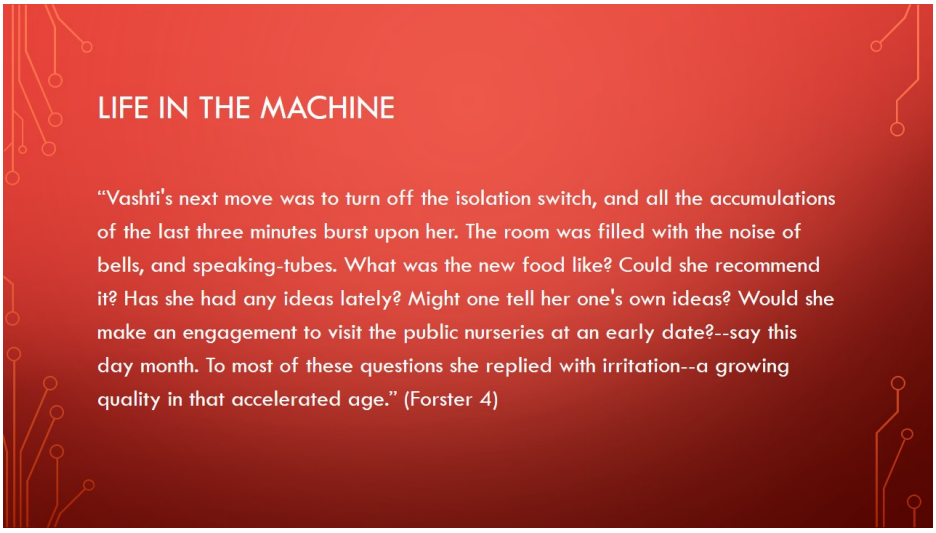
Slide 15



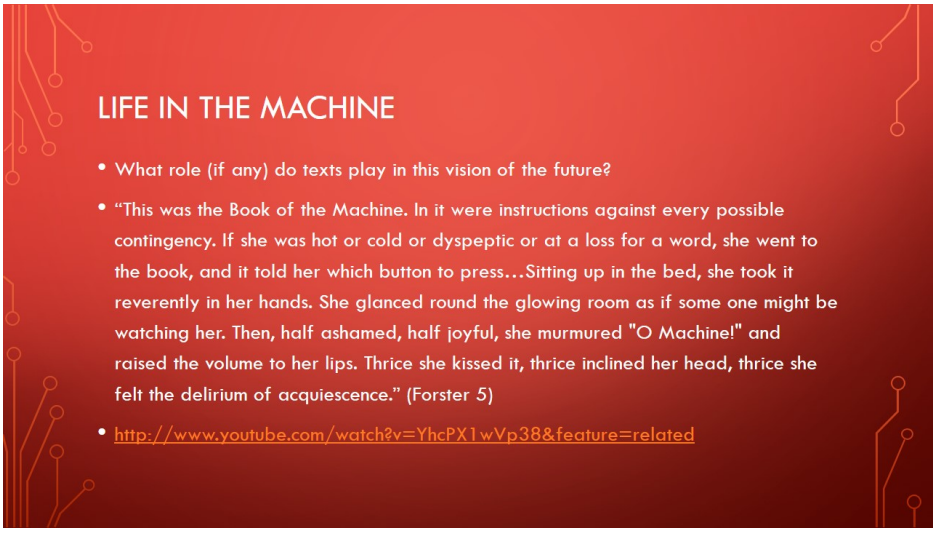
Slide 16



Slide 17



Slide 18



Slide 19

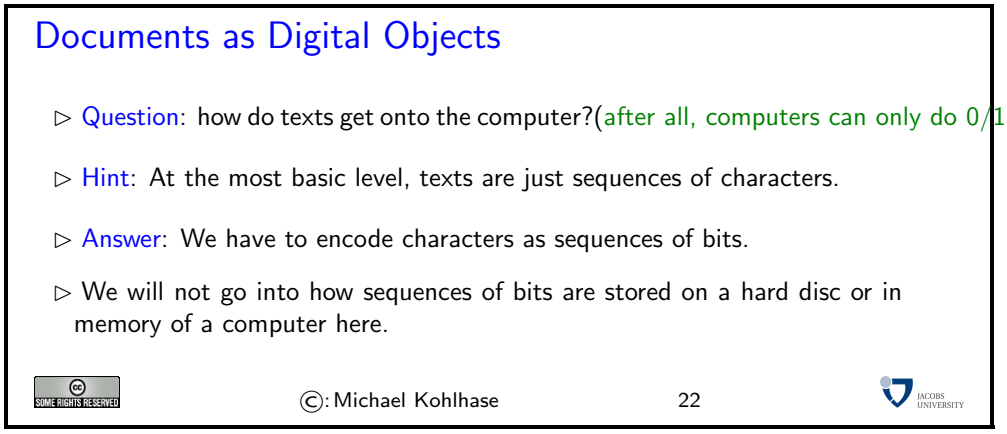
Slide 20


OUR EXPERIENCE AND RELATIONSHIP WITH TECHNOLOGY
• I learn about current events primarily online
• When reading for class, I prefer a print or hard copy of a book or article
• I have read at least one full-length book in digital form
• I am on at least one social networking site
• I have a cell phone that gives me access to the internet
• I have a blog
• I have my own website
• I have read at least one print book for leisure (not for class) in the past six months

Slide 21

Chapter 3

Documents as Digital Objects and their Meaning

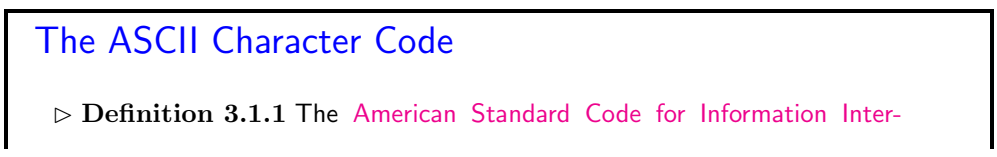


Before we go on, let us first get into some basics: how do we measure information, and how does this relate to units of information we know.

3.1 Character Codes in the Real World

We will now turn to a class of codes that are extremely important in information technology: character encodings. The idea here is that for IT systems we need to encode characters from our alphabets as bit strings (sequences of binary digits 0 and 1) for representation in computers. Indeed the Morse code we have seen above can be seen as a very simple example of a character encoding that is geared towards the manual transmission of natural languages over telegraph lines. For the encoding of written texts we need more extensive codes that can e.g. distinguish upper and lowercase letters.

The `ASCII` code we will introduce here is one of the first standardized and widely used character encodings for a complete alphabet. It is still widely used today. The code tries to strike a balance between a being able to encode a large set of characters and the representational capabilities in the time of punch cards (see below).



change (ASCII) code assigns characters to numbers 0-127

Code	...0	...1	...2	...3	...4	...5	...6	...7	...8	...9	...A	...B	...C	...D	...E	...F
0...	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1...	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2...	␣	!	"	#	$	%	&	'	(	)	*	+	,	−	.	/
3...	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4...	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5...	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6...	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7...	p	q	r	s	t	u	v	w	x	y	z	{	|	}	∼	DEL

The first 32 characters are control characters for ASCII devices like printers

⋙ Motivated by punchcards: The character 0 (binary 0000000) carries no information NUL, (used as dividers)
Character 127 (binary 1111111) can be used for deleting (overwriting) last value (cannot delete holes)

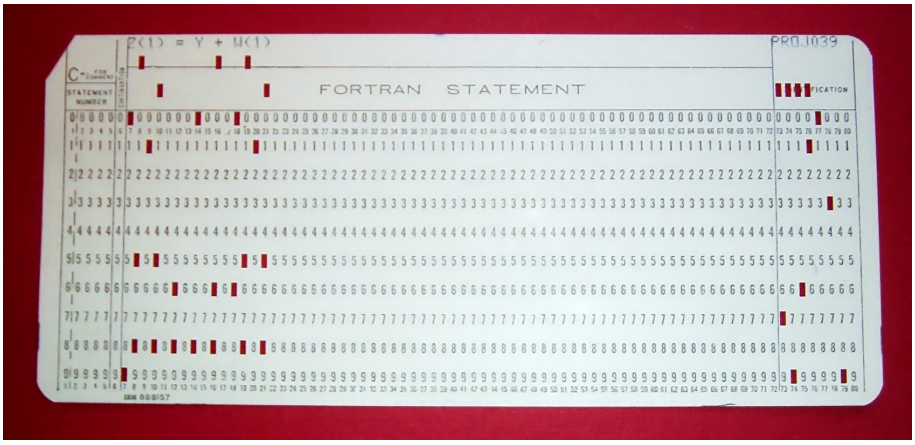
▷ The ASCII code was standardized in 1963 and is still prevalent in computers today (but seen as US-centric)

Punch cards were the the preferred medium for long-term storage of programs up to the late 1970s, since they could directly be produced by card punchers and automatically read by computers.

A Punchcard

▷ A punch card is a piece of stiff paper that contains digital information represented by the presence or absence of holes in predefined positions.

▷ **Example 3.1.2** This punch card encoded the FORTRAN statement `Z(1) = Y + W(1)`



Up to the 1970s, computers were batch machines, where the programmer delivered the program to the operator (a person behind a counter who fed the programs to the computer) and collected the

printouts the next morning. Essentially, each punch card represented a single line (80 characters) of program code. Direct interaction with a computer is a relatively young mode of operation.

The ASCII code as above has a variety of problems, for instance that the control characters are mostly no longer in use, the code is lacking many characters of languages other than the English language it was developed for, and finally, it only uses seven bits, where a byte (eight bits) is the preferred unit in information technology. Therefore there have been a whole zoo of extensions, which — due to the fact that there were so many of them — never quite solved the encoding problem.

Problems with ASCII encoding

- ▷ Problem: Many of the control characters are obsolete by now (e.g. NUL,BEL, or DEL)
- ▷ Problem: Many European characters are not represented (e.g. è,ñ,ü,ß,...)
- ▷ European ASCII Variants: Exchange less-used characters for national ones
- ▷ **Example 3.1.3 (German ASCII)** remap e.g. [↦ Ä,] ↦ Ü in German ASCII ("Apple][" comes out as "Apple ÜÄ")
- ▷ **Definition 3.1.4 (ISO-Latin (ISO/IEC 8859))** 16 Extensions of ASCII to 8-bit (256 characters) ISO-Latin 1 ≙ "Western European", ISO-Latin 6 ≙ "Arabic",ISO-Latin 7 ≙ "Greek"...
- ▷ Problem: No cursive Arabic, Asian, African, Old Icelandic Runes, Math,...
- ▷ Idea: Do something totally different to include all the world's scripts: For a scalable architecture, separate
  - ▷ what characters are available from the (character set)
  - ▷ bit string-to-character mapping (character encoding)

The goal of the UniCode standard is to cover all the worlds scripts (past, present, and future) and provide efficient encodings for them. The only scripts in regular use that are currently excluded are fictional scripts like the elvish scripts from the Lord of the Rings or Klingon scripts from the Star Trek series.

An important idea behind UniCode is to separate concerns between standardizing the character set — i.e. the set of encodable characters and the encoding itself.

Unicode and the Universal Character Set

- ▷ **Definition 3.1.5 (Twin Standards)** A scalable Architecture for representing all the worlds scripts
  - ▷ The Universal Character Set defined by the ISO/IEC 10646 International Standard, is a standard set of characters upon which many character encodings are based.
  - ▷ The Unicode Standard defines a set of standard character encodings, rules for normalization, decomposition, collation, rendering and bidirectional display order

▷ **Definition 3.1.6** Each UCS character is identified by an unambiguous name and an integer number called its code point.

▷ The UCS has 1.1 million code points and nearly 100 000 characters.

▷ **Definition 3.1.7** Most (non-Chinese) characters have code points in $[1, 65536]$ (the basic multilingual plane).

▷ **Notation 3.1.8** For code points in the Basic Multilingual Plane (BMP), four digits are used, e.g. U+0058 for the character LATIN CAPITAL LETTER X;

Note that there is indeed an issue with space-efficient encoding here. UniCode reserves space for 2^{32} (more than a million) characters to be able to handle future scripts. But just simply using 32 bits for every UniCode character would be extremely wasteful: UniCode-encoded versions of `ASCII` files would be four times as large.

Therefore UniCode allows multiple encodings. UTF-32 is a simple 32-bit code that directly uses the code points in binary form. UTF-8 is optimized for western languages and coincides with the `ASCII` where they overlap. As a consequence, `ASCII` encoded texts can be decoded in UTF-8 without changes — but in the UTF-8 encoding, we can also address all other UniCode characters (using multi-byte characters).

Character Encodings in Unicode

▷ **Definition 3.1.9** A character encoding is a mapping from bit strings to UCS code points.

▷ Idea: Unicode supports multiple encodings (but not character sets) for efficiency

▷ **Definition 3.1.10 (Unicode Transformation Format)** ▷ UTF-8, 8-bit, variable-width encoding, which maximizes compatibility with ASCII.

▷ UTF-16, 16-bit, variable-width encoding (popular in Asia)

▷ UTF-32, a 32-bit, fixed-width encoding (for safety)

▷ **Definition 3.1.11** The UTF-8 encoding follows the following encoding scheme

Unicode	Byte1	Byte2	Byte3	Byte4
U+000000 – U+00007F	0xxxxxxx			
U+000080 – U+0007FF	110xxxxx	10xxxxxx		
U+000800 – U+00FFFF	1110xxxx	10xxxxxx	10xxxxxx	
U+010000 – U+10FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

▷ **Example 3.1.12** $\$ =$ U+0024 is encoded as 00100100 (1 byte)

¢ = U+00A2 is encoded as 11000010,10100010 (two bytes)

$e =$ U+20AC is encoded as 11100010,10000010,10101100 (three bytes)

Note how the fixed bit prefixes in the encoding are engineered to determine which of the four cases apply, so that UTF-8 encoded documents can be safely decoded..

3.2 Measuring Sizes of Digital Documents

How much Information?

Bit (b)	*binary digit 0/1*
Byte (B)	*8 bit*
2 Bytes	A Unicode character in UTF.
10 Bytes	your name.
Kilobyte (kB)	*1,000 bytes OR* 10^3 *bytes*
2 Kilobytes	A Typewritten page.
100 Kilobytes	A low-resolution photograph.
Megabyte (MB)	*1,000,000 bytes OR* 10^6 *bytes*
1 Megabyte	A small novel or a 3.5 inch floppy disk.
2 Megabytes	A high-resolution photograph.
5 Megabytes	The complete works of Shakespeare.
10 Megabytes	A minute of high-fidelity sound.
100 Megabytes	1 meter of shelved books.
500 Megabytes	A CD-ROM.
Gigabyte (GB)	*1,000,000,000 bytes or* 10^9 *bytes*
1 Gigabyte	a pickup truck filled with books.
20 Gigabytes	A good collection of the works of Beethoven.
100 Gigabytes	A library floor of academic journals.

Terabyte (TB)	*1,000,000,000,000 bytes or* 10^{12} *bytes*
1 Terabyte	50000 trees made into paper and printed.
2 Terabytes	An academic research library.
10 Terabytes	The print collections of the U.S. Library of Congress.
400 Terabytes	National Climate Data Center (NOAA) database.
Petabyte (PB)	*1,000,000,000,000,000 bytes or* 10^{15} *bytes*
1 Petabyte	3 years of EOS data (2001).
2 Petabytes	All U.S. academic research libraries.
20 Petabytes	Production of hard-disk drives in 1995.
200 Petabytes	All printed material (ever).
Exabyte (EB)	*1,000,000,000,000,000,000 bytes or* 10^{18} *bytes*
2 Exabytes	Total volume of information generated in 1999.
5 Exabytes	All words ever spoken by human beings ever.
300 Exabytes	All data stored digitally in 2007.
Zettabyte (ZB)	*1,000,000,000,000,000,000,000 bytes or* 10^{21} *bytes*
2 Zettabytes	Total volume digital data transmitted in 2011
100 Zettabytes	Data equivalent to the human Genome in one body.

The information in this table is compiled from various studies, most recently [HL11].

Note: Information content of real-world artifacts can be assessed differently, depending on the view. Consider for instance a text typewritten on a single page. According to our definition, this has ca. 2 kB, but if we fax it, the image of the page has 2 MB or more, and a recording of a text read out loud is ca. 50 MB. Whether this is a terrible waste of bandwidth depends on the

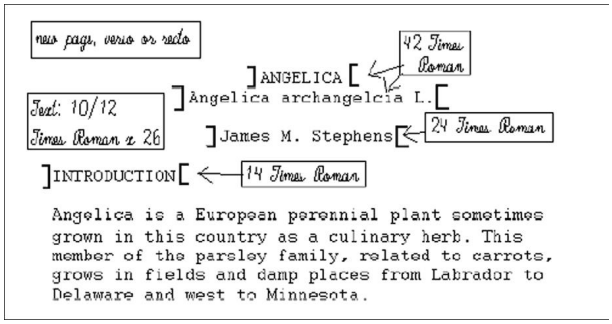
application. On a fax, we can use the shape of the signature for identification (here we actually care more about the shape of the ink mark than the letters it encodes) or can see the shape of a coffee stain. In the audio recording we can hear the inflections and sentence melodies to gain an impression on the emotions that come with text.

3.3 Texts are more than Sequences of Characters

Document Markup

▷ **Definition 3.3.1 (Document Markup)** Document markupmarkup is the process of adding codes (special, standardized character sequences) to a document to control the structure, formatting, or the relationship among its parts.

▷ **Example 3.3.2** A text with markup codes (for printing)



There are many systems for document markup ranging from informal ones as in Definition 7.3.1 that specify the intended document appearance to humans – in this case the printer – to technical ones which can be understood by machines but serving the same purpose.

Styles of Markup

▷ **Definition 3.3.3 (Presentation Markup)** A presentation markup scheme is one that specifies document structure to aid document processing by humans

▷ **Example 3.3.4** e.g. *roff, Postscript, DVI, early MS Word, low-level TEX

\+ simple, context-free, portable (verbatim), easy to implement/transform

– inflexible, possibly verbose,

▷ **Definition 3.3.5 (Content Markup)** A content markup scheme is one that specifies document structure to aid document processing by machines or with machine support.

▷ **Example 3.3.6** e.g. LATEX (if used correctly), Programming Languages, ATP input

+ flexible, portable (in spirit), unambiguous, language-independent
– possibly verbose, context dependent, hard to read and write

Content vs. Presentation by Example

Format	Representation	Content?
LaTeX	`{\textbf{proof}}:...\hfill\Box`	`\begin{proof}...\end{proof}`
HTML	`<font size=+2><b>...</b></font>`	`<h1>...</h1>`
Lisp	$8+\sqrt{x^3}$	`(power (plus 8 (sqrt x)) 3)`
TeX	`$\{f\|f(0)> 0{\rm and}f(1)<0\}$`	$\{f\|f(0) > 0 \text{ and } f(1) < 0\}$
TeX	`$\{f\|f(0)> 0$ and $f(1)<0\}$`	$\{f\|f(0) > 0 \text{ and } f(1) < 0\}$

▷ We consider these to be representations of the same content (object)

▷ Problem: Transformations between presentation and content Markup

▷ Content $\rightsquigarrow$ Pres.: usually done by styling (++ user-adaptivity)

▷ Pres. $\rightsquigarrow$ Content: Heuristic Process(e.g. binomials $\binom{n}{k}$ vs. C_k^n vs. C_n^k)

Content vs. Semantics/Formalization

▷ Content: logic-independent infrastructure
Identification of abstract syntax, "semantics" by reference for symbols.

```
<apply>
  <plus/>
  <csymbol definitionURL="mbase://numbers/perfect#the-smallest"/>
  <cn>2</cn>
</apply>
```

▷ Semantics: establishing meaning by fixing consequences
adds formal inference rules and axioms.

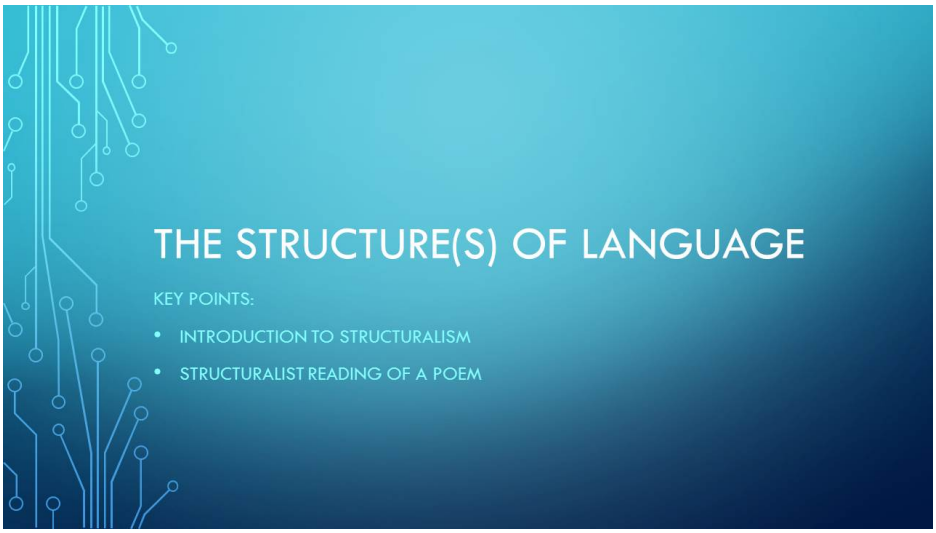
▷ Mechanization in a specific system (Thm Prover or Proof Checker)

▷ logical framework (specify the logic in the system itself)

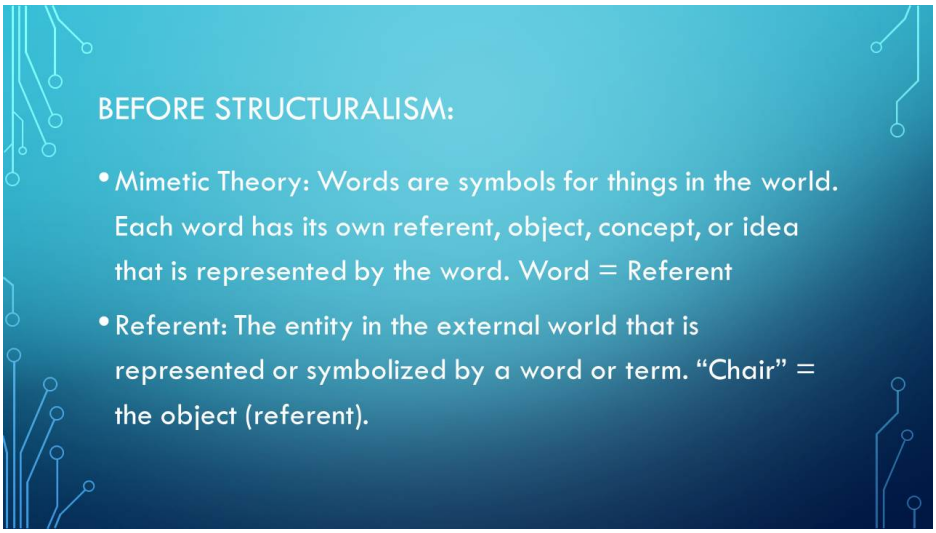
Chapter 4

Documents and Meaning

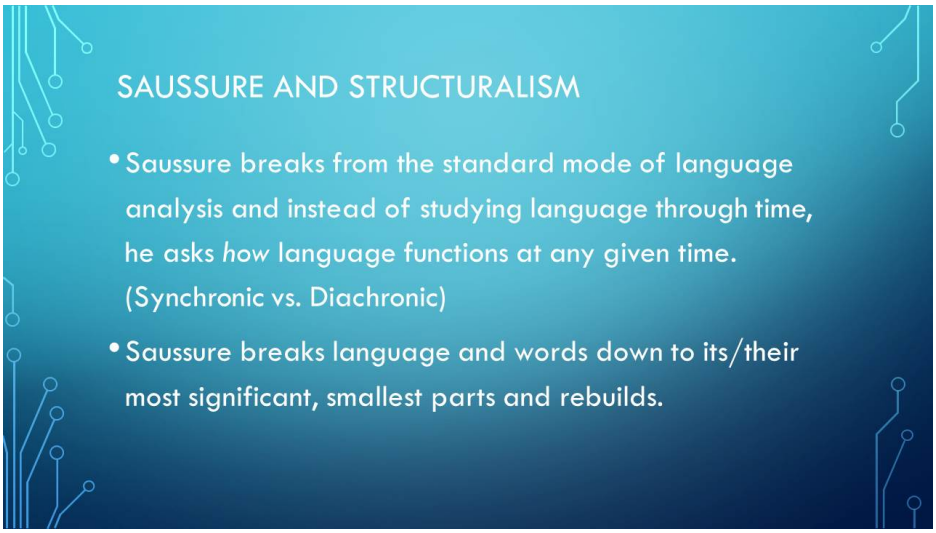
4.1 Structuralism



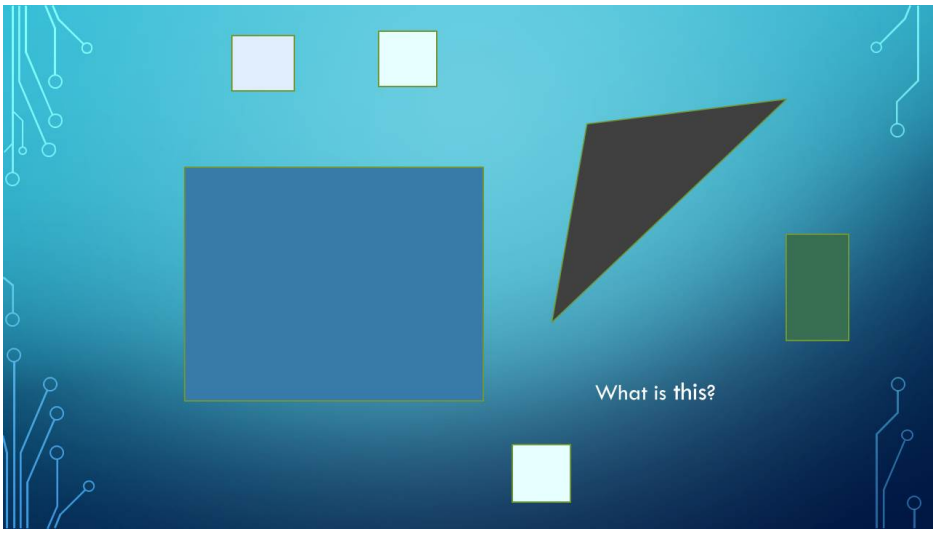
Slide 33



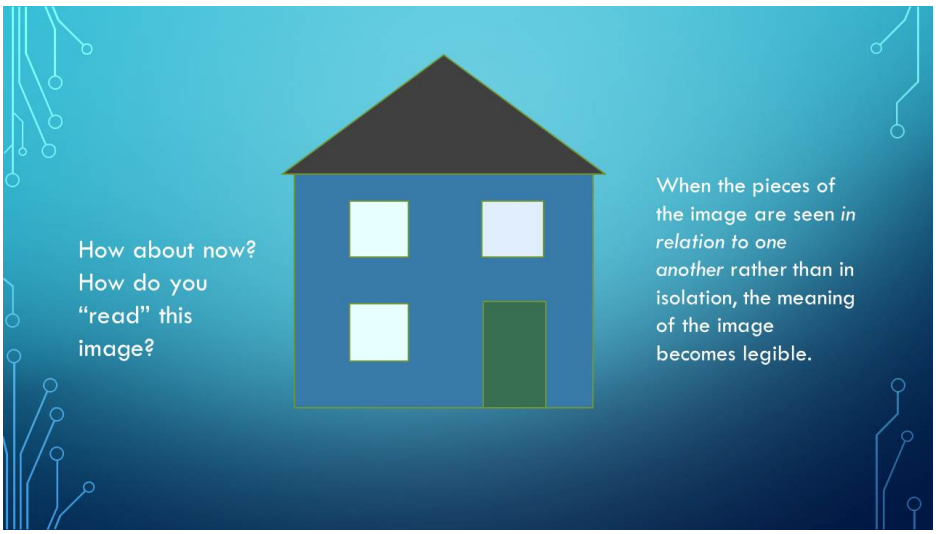
Slide 34



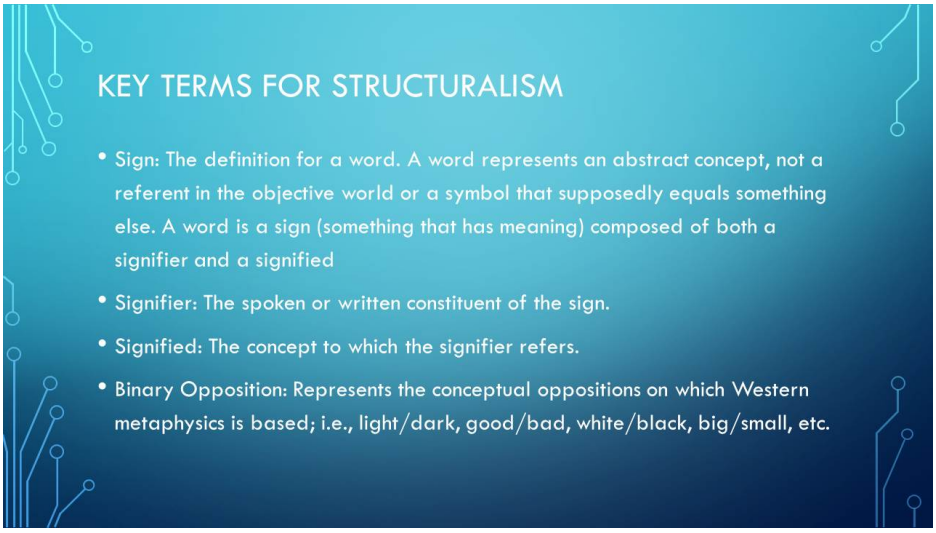
Slide 35



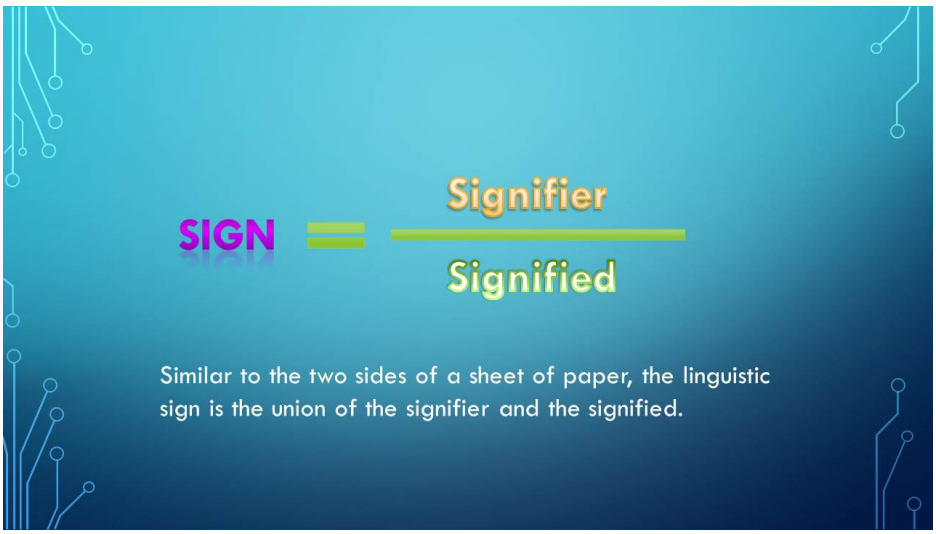
Slide 36



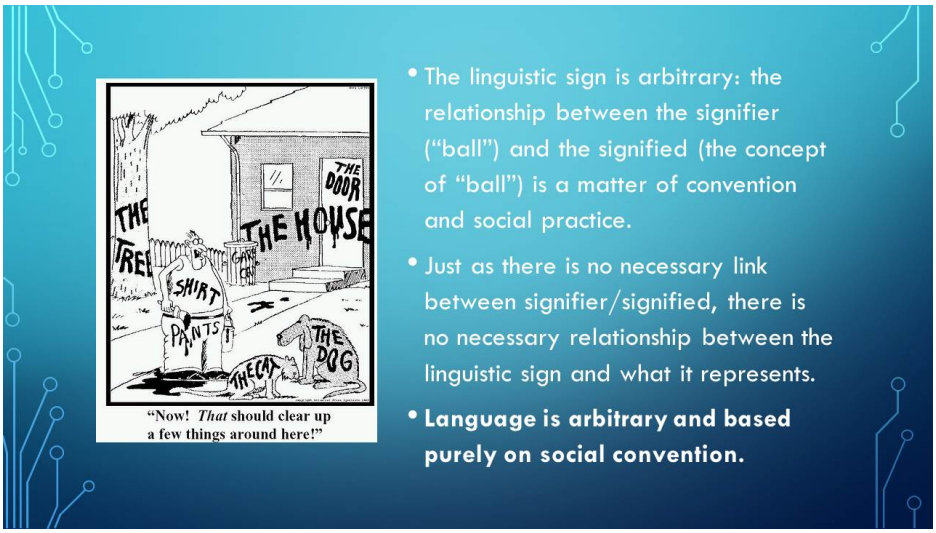
Slide 37



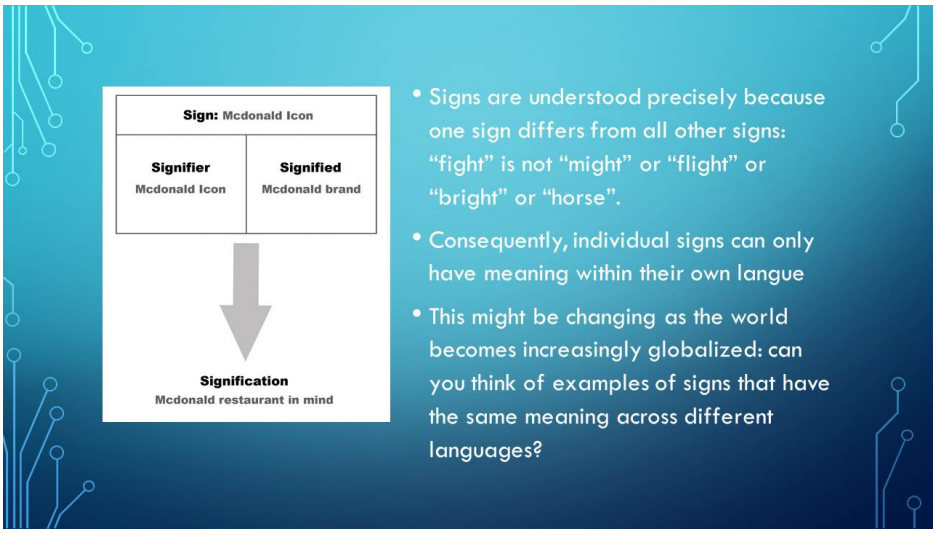
Slide 38



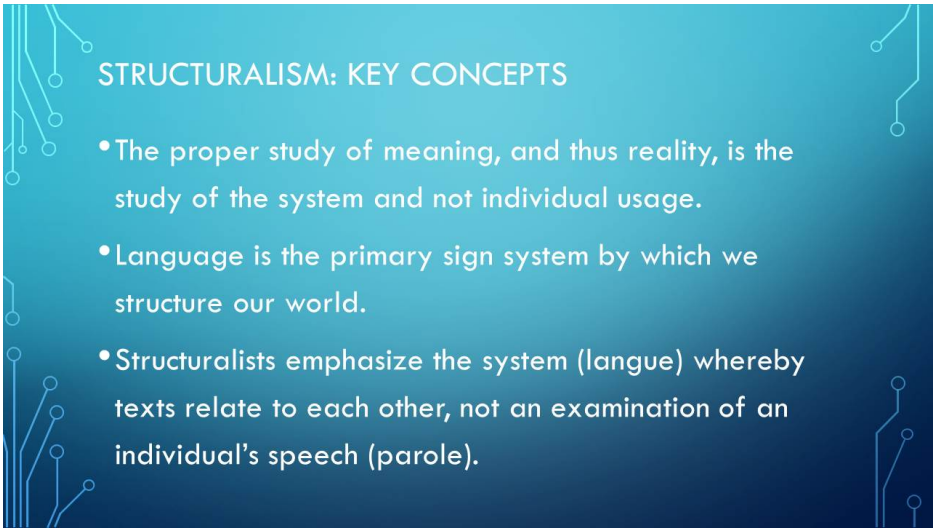
Slide 39



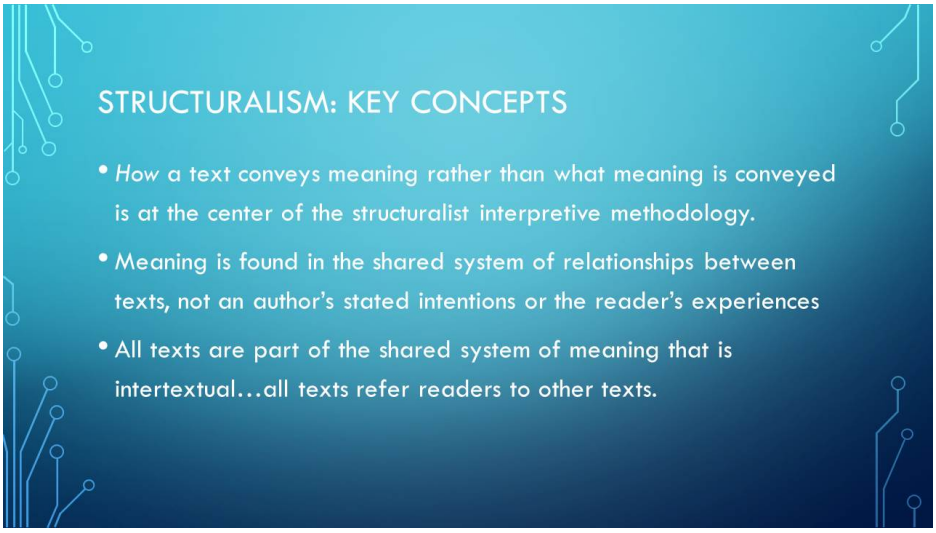
Slide 40



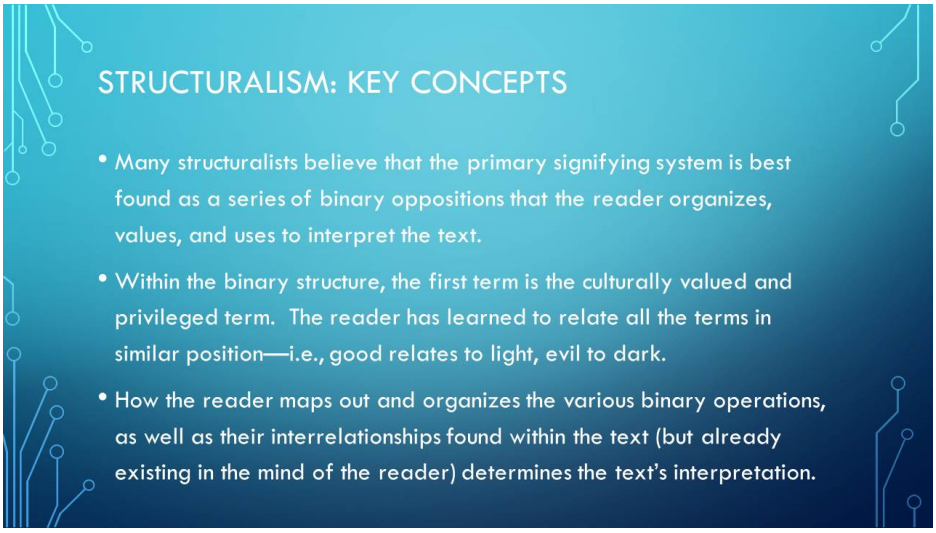
Slide 41



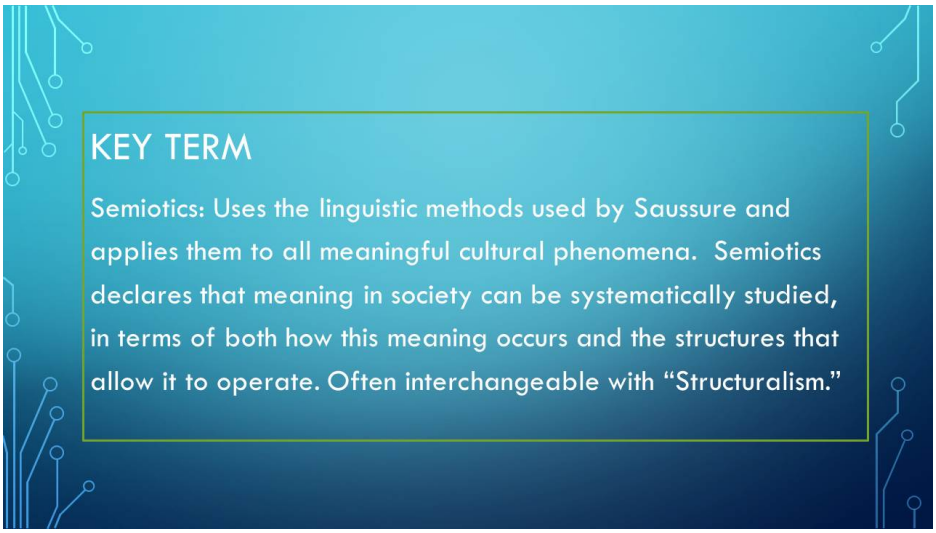
Slide 42



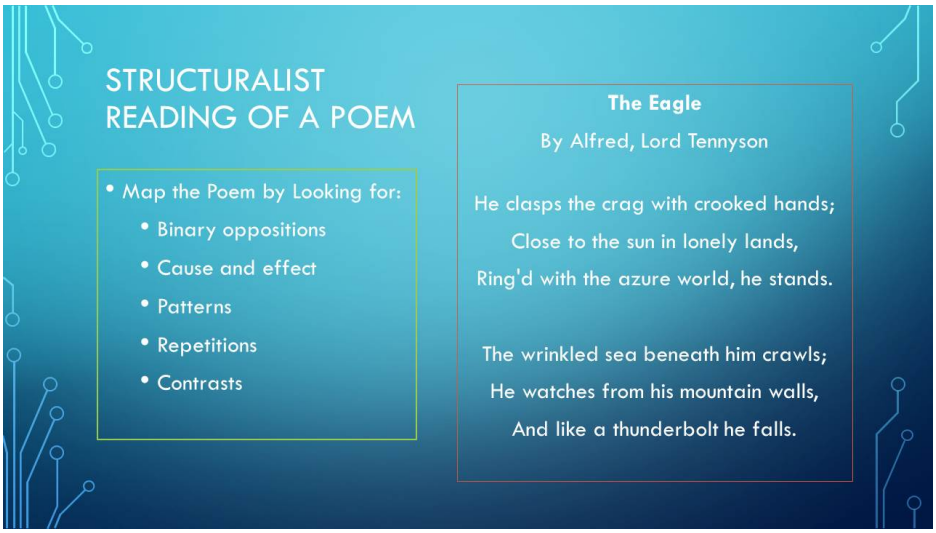
Slide 43



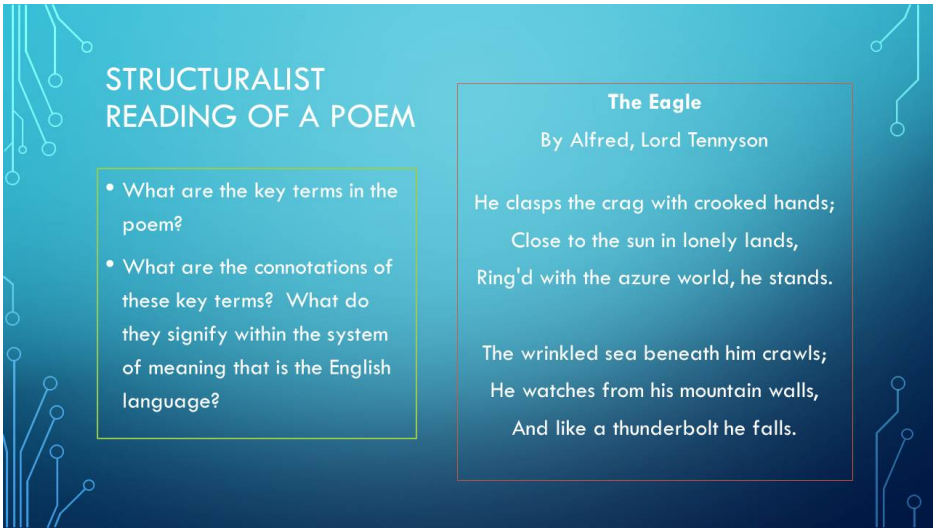
Slide 44



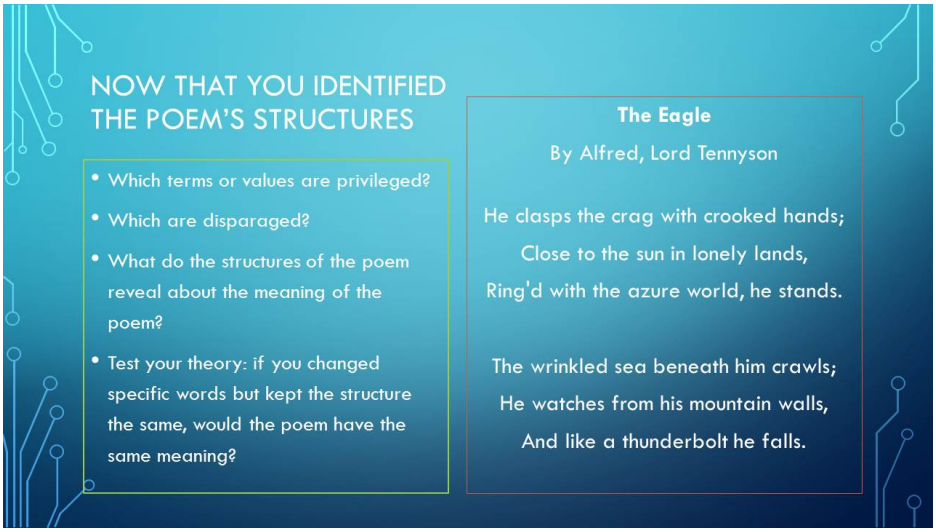
Slide 45



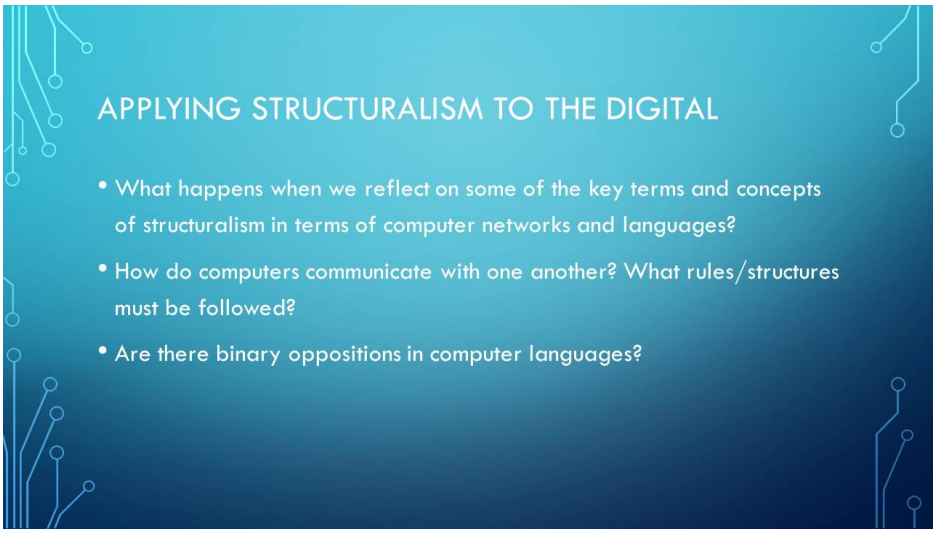
Slide 46



Slide 47



Slide 48



Slide 49

4.2 Formal Logic as the Mathematics of Meaning

What a logic is and is good for?

▷ Q: What is a "logic"?

▷ A: A mathematical language that can describe non-mathematical and mathematical facts in mathematical "propositions".

▷ Q: What can you "do" with a logic?

▷ A: You can formally derive ("deduce") new propositions from ones that

you already have, thereby answering interesting questions about the reality you are describing.

▷ You already know Logic (Propositional Logic)

▷ We will learn about a powerful extension: First-Order Logic

What is Logic?

▷ formal languages, inference and their relation with the world

- ▷ Formal language $\mathcal{FL}$: set of formulae $(2+3/7,\ \forall x.x+y=y+x)$
- ▷ Formula: sequence/tree of symbols $(x, y, f, g, p, 1, \pi, \in, \neg, \wedge \forall, \exists)$
- ▷ Models: things we understand (e.g. number theory)
- ▷ Interpretation: maps formulae into models $([\![\text{three plus five}]\!] = 8)$
- ▷ Validity: $\mathcal{M} \models \mathbf{A}$, iff $[\![\mathbf{A}]\!]^{\mathcal{M}} = \mathsf{T}$ (five greater three is valid)
- ▷ Entailment: $\mathbf{A} \models \mathbf{B}$, iff $\mathcal{M} \models \mathbf{B}$ for all $\mathcal{M} \models \mathbf{A}$.(generalize to $\mathcal{H} \models \mathbf{A}$)
- ▷ Inference: rules to transform (sets of) formulae $(\mathbf{A}, \mathbf{A} \Rightarrow \mathbf{B} \vdash \mathbf{B})$

▷ Syntax: formulae, inference (just a bunch of symbols)

▷ Semantics: models, interpr., validity, entailment (math. structures)

▷ Important Question: relation between syntax and semantics?

So logic is the study of formal representations of objects in the real world, and the formal statements that are true about them. The insistence on a *formal language* for representation is actually something that simplifies life for us. Formal languages are something that is actually easier to understand than e.g. natural languages. For instance it is usually decidable, whether a string is a member of a formal language. For natural language this is much more difficult: there is still no program that can reliably say whether a sentence is a grammatical sentence of the English language.

We have already discussed the meaning mappings (under the monicker "semantics"). Meaning mappings can be used in two ways, they can be used to understand a formal language, when we use a mapping into "something we already understand", or they are the mapping that legitimize a representation in a formal language. We understand a formula (a member of a formal language) $\mathbf{A}$ to be a representation of an object $\mathcal{O}$, iff $[\![\mathbf{A}]\!] = \mathcal{O}$.

However, the game of representation only becomes really interesting, if we can do something with the representations. For this, we give ourselves a set of syntactic rules of how to manipulate the formulae to reach new representations or facts about the world.

Consider, for instance, the case of calculating with numbers, a task that has changed from a difficult job for highly paid specialists in Roman times to a task that is now feasible for young children. What is the cause of this dramatic change? Of course the formalized reasoning procedures for arithmetic that we use nowadays. These *calculi* consist of a set of rules that can be followed purely syntactically, but nevertheless manipulate arithmetic expressions in a correct and fruitful way. An essential prerequisite for syntactic manipulation is that the objects are given in a formal language suitable for the problem. For example, the introduction of the decimal system has been

instrumental to the simplification of arithmetic mentioned above. When the arithmetical calculi were sufficiently well-understood and in principle a mechanical procedure, and when the art of clock-making was mature enough to design and build mechanical devices of an appropriate kind, the invention of calculating machines for arithmetic by Wilhelm Schickard (1623), Blaise Pascal (1642), and Gottfried Wilhelm Leibniz (1671) was only a natural consequence.

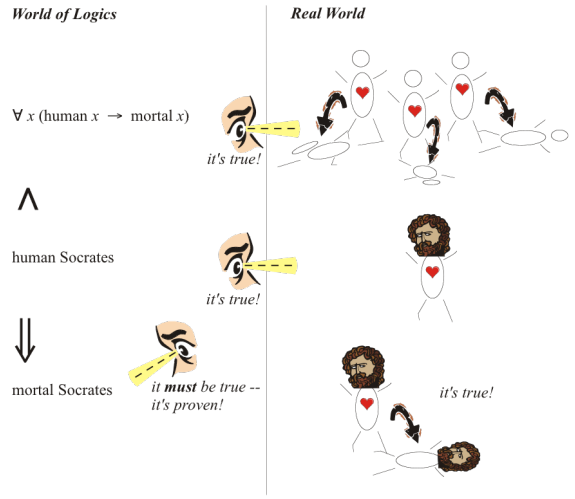
We will see that it is not only possible to calculate with numbers, but also with representations of statements about the world (propositions). For this, we will use an extremely simple example; a fragment of propositional logic (we restrict ourselves to only one logical connective) and a small calculus that gives us a set of rules how to manipulate formulae.

Within the world of logics, one can derive new propositions (the *conclusions*, here: *Socrates is mortal*) from given ones (the *premises*, here: *Every human is mortal* and *Sokrates is human*). Such derivations are *proofs*.

In particular, logics can describe the internal structure of real-life facts; e.g. individual things, actions, properties. A famous example, which is in fact as old as it appears, is illustrated in the slide below.

The miracle of logics

▷ Purely formal derivations are true in the real world!



If a logic is correct, the conclusions one can prove are true (= hold in the real world) whenever the premises are true. This is a miraculous fact (think about it!)

In general formulae can be used to represent facts about the world as propositions; they have a semantics that is a mapping of formulae into the real world (propositions are mapped to truth values.) We have seen two relations on formulae: the entailment relation and the deduction relation. The first one is defined purely in terms of the semantics, the second one is given by a calculus, i.e. purely syntactically. Is there any relation between these relations?

Soundness and Completeness

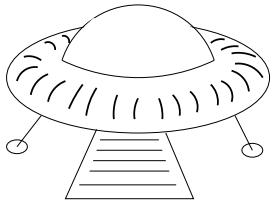
▷ **Definition 4.2.1** Let $\mathcal{S} := \langle \mathcal{L}, \mathcal{K}, \models \rangle$ be a logical system, then we call a calculus $\mathcal{C}$ for $\mathcal{S}$

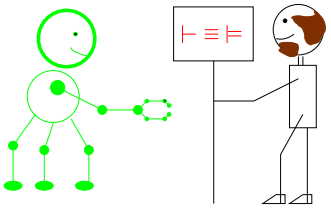
▷ sound (or correct), iff $\mathcal{H} \models \mathbf{A}$, whenever $\mathcal{H} \vdash_{\mathcal{C}} \mathbf{A}$, and

▷ complete, iff $\mathcal{H} \vdash_{\mathcal{C}} \mathbf{A}$, whenever $\mathcal{H} \models \mathbf{A}$.

▷ Goal: $\vdash \mathbf{A}$ iff $\models \mathbf{A}$ (provability and validity coincide)

▷ To TRUTH through PROOF (CALCULEMUS [Leibniz ∼1680])





Ideally, both relations would be the same, then the calculus would allow us to infer all facts that can be represented in the given formal language and that are true in the real world, and only those. In other words, our representation and inference is faithful to the world.

A consequence of this is that we can rely on purely syntactical means to make predictions about the world. Computers rely on formal representations of the world; if we want to solve a problem on our computer, we first represent it in the computer (as data structures, which can be seen as a formal language) and do syntactic manipulations on these structures (a form of calculus). Now, if the provability relation induced by the calculus and the validity relation coincide (this will be quite difficult to establish in general), then the solutions of the program will be correct, and we will find all possible ones.

Three Principal Modes of Inference

▷ Deduction: knowledge extension $\dfrac{rains \Rightarrow wet\_street \quad rains}{wet\_street} D$

▷ Abduction explanation $\dfrac{rains \Rightarrow wet\_street \quad wet\_street}{rains} A$

▷ Induction learning rules $\dfrac{wet\_street \quad rains}{rains \Rightarrow wet\_street} I$

4.3 Using Logic to Model Meaning of Natural Language

Modeling Natural Language Semantics

▷ Problem: Find formal (logic) system for the meaning of natural language

▷ History of ideas

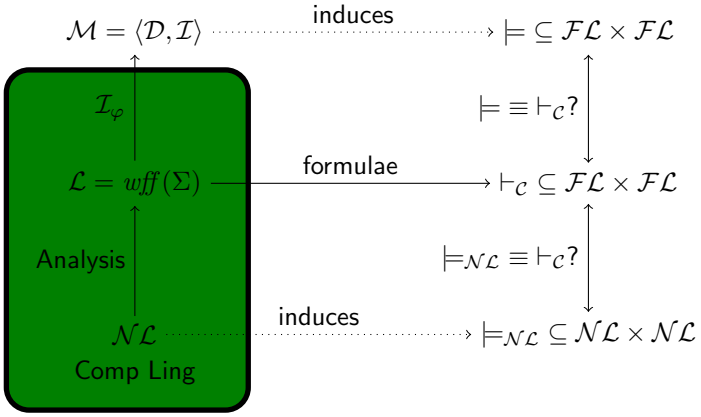
▷ Propositional logic [ancient Greeks like Aristotle]
Every human is mortal

- First-Order Predicate logic [Frege ≤ 1900]
  I believe, that my audience already knows this.
- Modal logic [Lewis18, Kripke65]
  A man sleeps. He snores. $((\exists X.\mathsf{man}(X) \wedge \mathsf{sleep}(X))) \wedge \mathsf{snore}(X)$
- Various dynamic approaches (e.g. DRT, DPL)
  Most men wear black
- Higher-order Logic, e.g. generalized quantifiers
- ...

 ©: Michael Kohlhase 55

Let us now reconcider the role of all of this for natural language semantics. We have claimed that the goal of the course is to provide you with a set of methods to determine the meaning of natural language. If we look back, all we did was to establish translations from natural languages into formal languages like first-order or higher-order logic (and that is all you will find in most semantics papers and textbooks). Now, we have just tried to convince you that these are actually syntactic entities. So, *where is the semantics?*.

Natural Language Semantics?



 ©: Michael Kohlhase 56

As we mentioned, the green area is the one generally covered by natural language semantics. In the analysis process, the natural language utterances (viewed here as formulae of a language $\mathcal{NL}$) are translated to a formal language $\mathcal{FL}$ (a set $\mathit{wff}(\Sigma)$ of well-formed formulae). We claim that this is all that is needed to recapture the semantics even it this is not immediately obvious at first: Theoretical Logic gives us the missing pieces.

Since $\mathcal{FL}$ is a formal language of a logical systems, it comes with a notion of model and an interpretation function $\mathcal{I}_\varphi$ that translates $\mathcal{FL}$ formulae into objects of that model. This induces a notion of logical consequence[1] as explained in[7]. It also comes with a calculus $\mathcal{C}$ acting on $\mathcal{FL}$-formulae, which (if we are lucky) is correct and complete (then the mappings in the upper rectangle commute). EdN:7

What we are really interested in in natural language semantics is the truth conditions and natural consequence relations on natural language utterances, which we have denoted by $\models_{\mathcal{NL}}$.

[1] Relations on a set S are subsets of the cartesian product of S, so we use $R \in (S^*)S$ to signify that R is a (n-ary) relation on X.

[7] EdNote: crossref

If the calculus $\mathcal{C}$ of the logical system $\langle \mathcal{FL}, \mathcal{K}, \models \rangle$ is adequate (it might be a bit presumptious to say sound and complete), then it is a model of the relation $\models_{\mathcal{NL}}$. Given that both rectangles in the diagram commute, then we really have a model for truth-conditions and logical consequence for natural language utterances, if we only specify the analysis mapping (the green part) and the calculus.

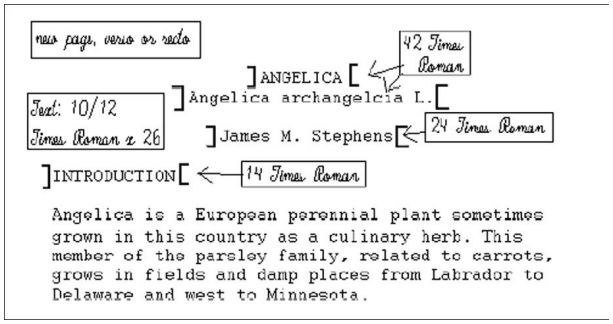
Logic-Based Knowledge Representation for NLP

- ▷ Logic (and related formalisms) allow to integrate world knowledge
  - ▷ explicitly (gives more understanding than statistical methods)
  - ▷ transparently (symbolic methods are monotonic)
  - ▷ systematically (we can prove theorems about our systems)
- ▷ Signal + World knowledge makes more powerful model
  - ▷ Does not preclude the use of statistical methods to guide inference
- ▷ Problems with logic-based approaches
  - ▷ Where does the world knowledge come from? (Ontology problem)
  - ▷ How to guide search induced by log. calculi (combinatorial explosion)

One possible answer: Description Logics. (next couple of times)

▷ Document Markup

- ▷ **Definition 4.3.1 (Document Markup)** Document markupmarkup is the process of adding codes (special, standardized character sequences) to a document to control the structure, formatting, or the relationship among its parts.
- ▷ **Example 4.3.2** A text with markup codes (for printing)

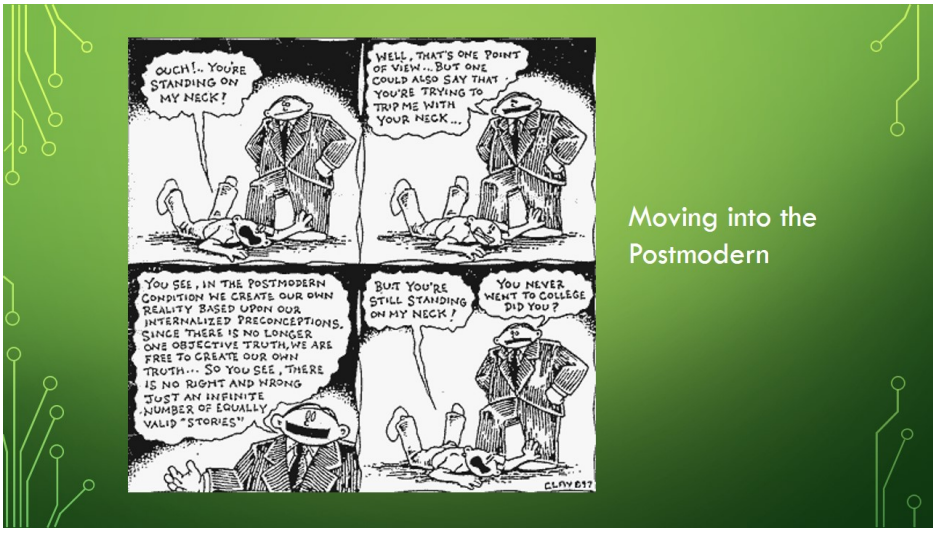


There are many systems for document markup ranging from informal ones as in Definition 7.3.1 that specify the intended document appearance to humans – in this case the printer – to technical ones which can be understood by machines but serving the same purpose.

4.4 Deconstruction

we start with the first stlide

Slide 59



Slide 60

MOVING INTO THE POSTMODERN

- There is no truth; truth is only and always subjective
- Truth is always a construct(ion)
- Truth is always relative so there are always and simultaneously many truths
- Since truth is always relative, there is no objective reality
- So all meaning is constructed through difference (there is no center)
- And that means meaning itself is always subjective and in flux

Slide 61

An optical illusion might also help illustrate the key ideas of postmodernism:

- Is this the image of an old woman or the image of a young woman?
- What determines how we view the image?

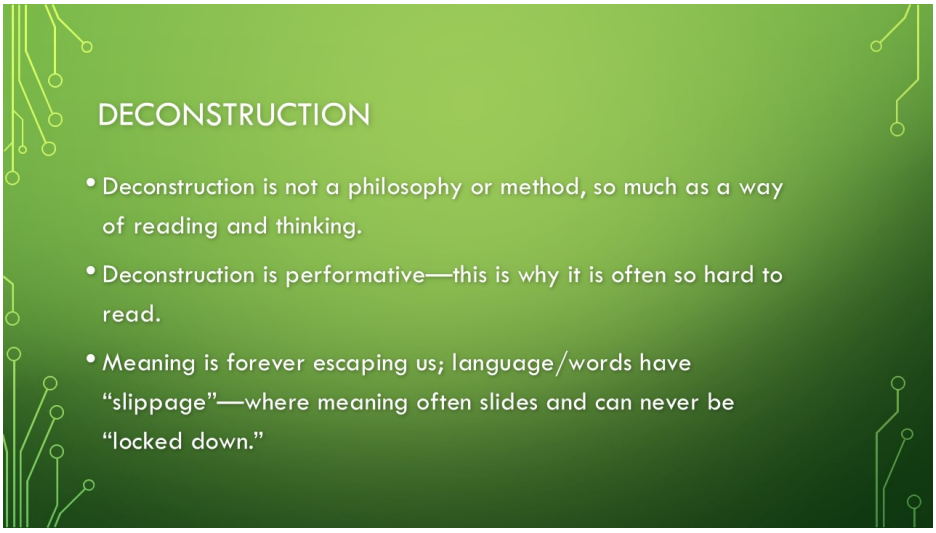
Slide 62

BUILDING FROM STRUCTURALISM

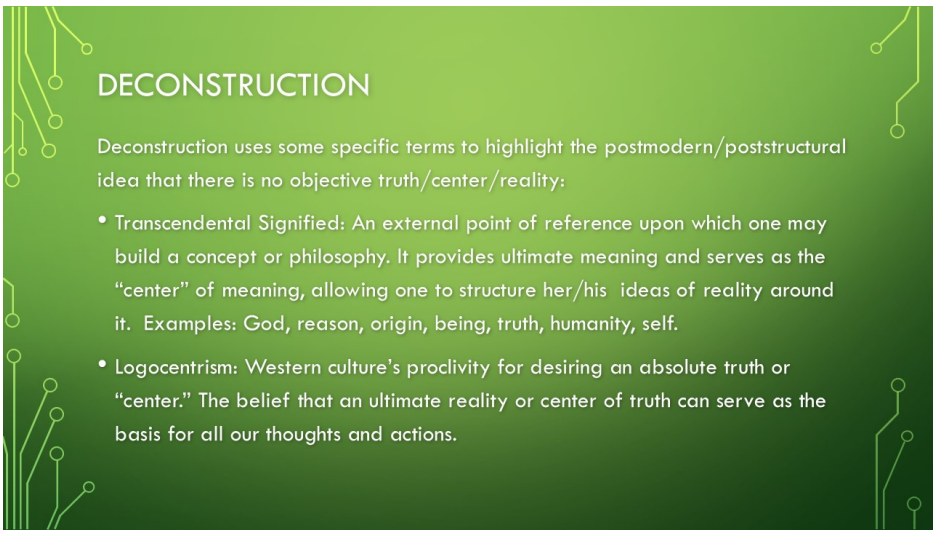
Derrida builds off of Saussure and structuralism in a few key areas:

- Meaning in language is constructed through the differences between/among signs
- Emphasis on intertextuality
- Binary oppositions are a fundamental structure within language and thought

Slide 63



Slide 64



Slide 65

DECONSTRUCTION: BINARY OPPOSITIONS

- Supplement: The unstable relationship between the two elements contained in a binary. Rather than being two distinct ideas/concepts, each informs the other. Each term helps define the other and is necessary for the other to exist.
- Différance: Two simultaneous meanings: 1) To defer, postpone, or delay; 2) to differ, to be different from. Caught in an endless chain of signifiers: each signifier differs from another, but only defers the other meaning.
- This is the essence of différance: meaning always is deferred and differing.

Slide 66

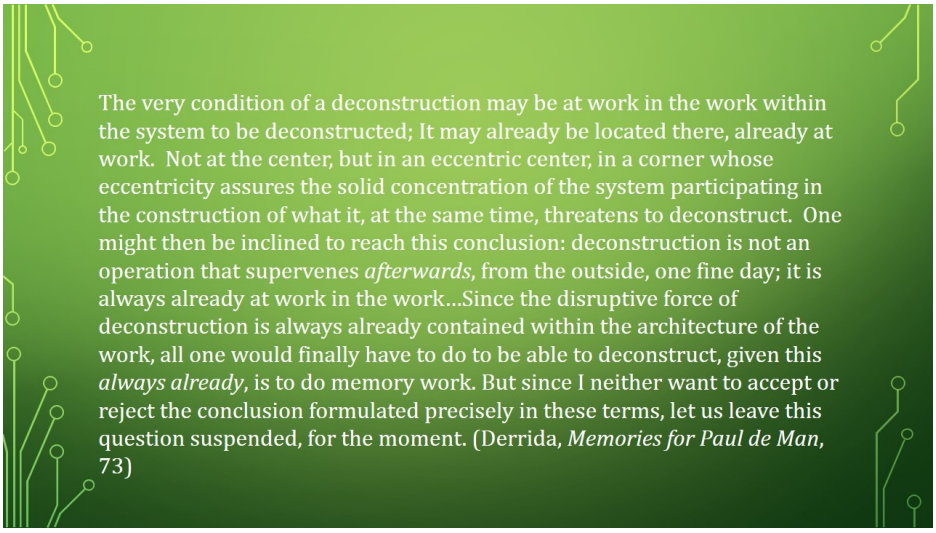
DECONSTRUCTION: BINARY OPPOSITIONS

- Deconstruction is interested in pulling apart, exploring and questions these binary oppositions
- Derrida does not believe we can get outside the binary (we are trapped in that deeply habitual way of thinking) but we can question the structure, question the binary itself by reversing the hierarchy
- Why would we do this? The goal is to gain new or more insight/knowledge into our own thinking and ways we create meaning(s)

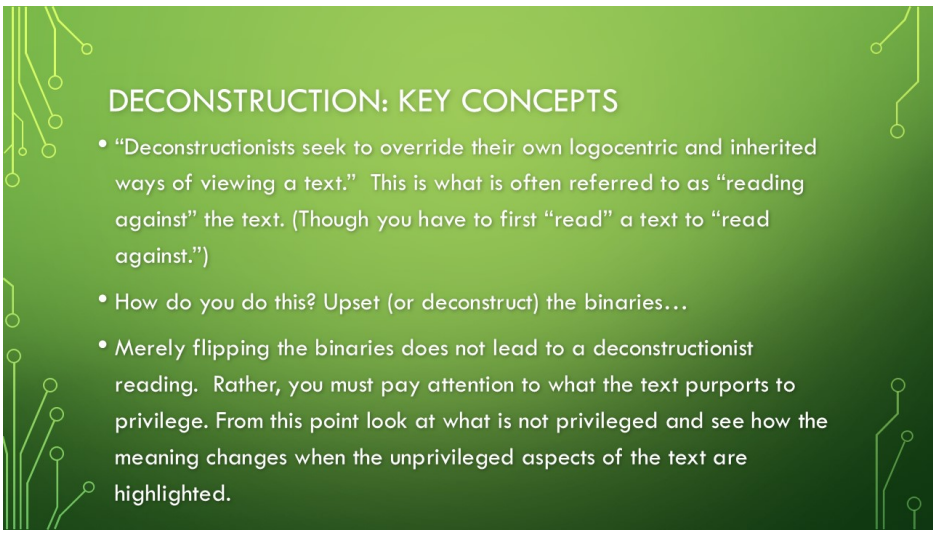
Slide 67

Derrida on Deconstruction:
https://www.youtube.com/watch?v=vgwOjjoYtco

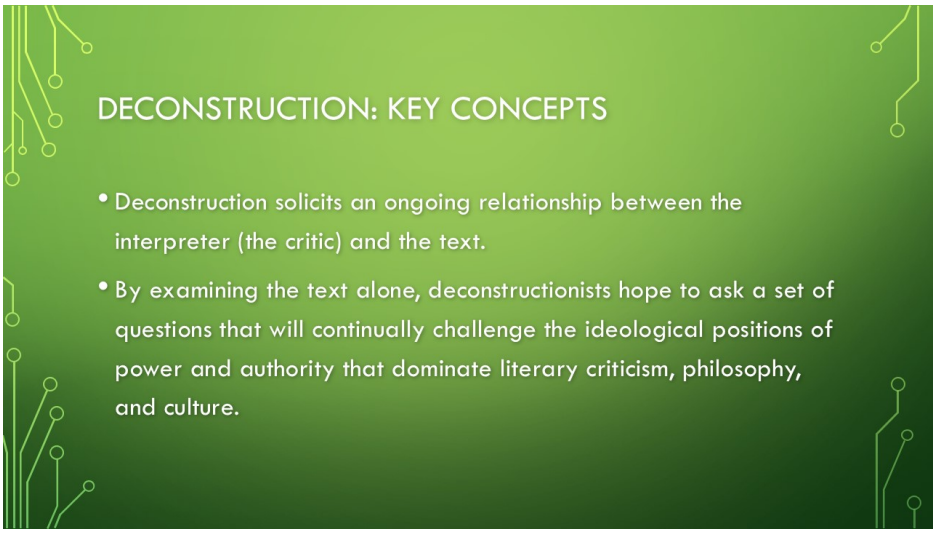
Slide 68



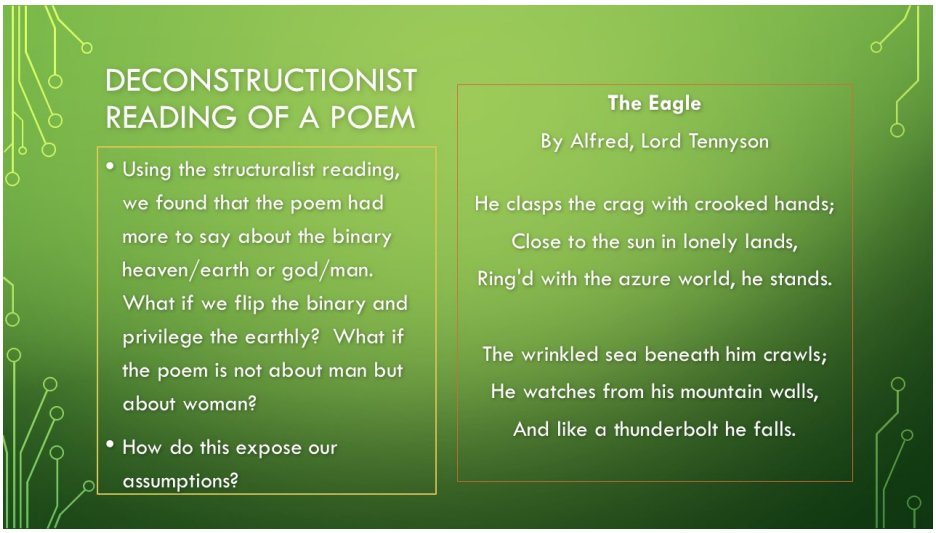
Slide 69



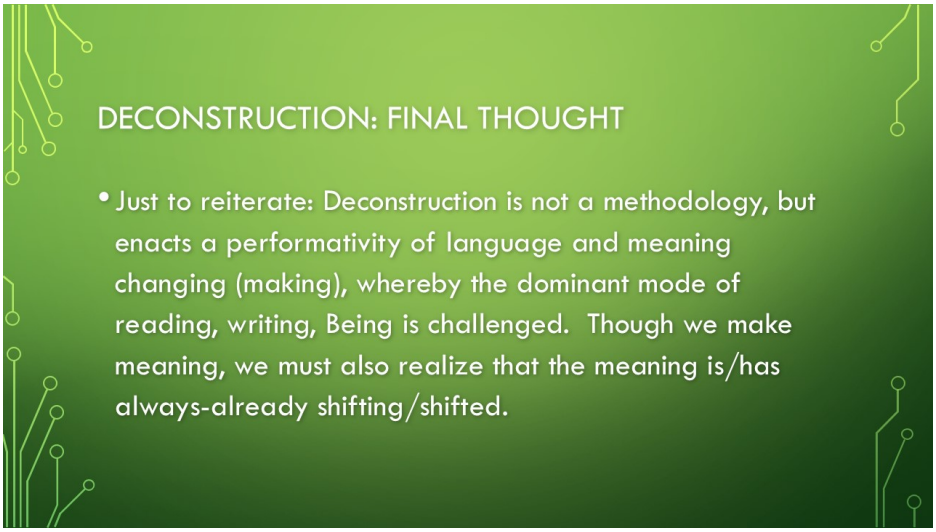
Slide 70



Slide 71



Slide 72

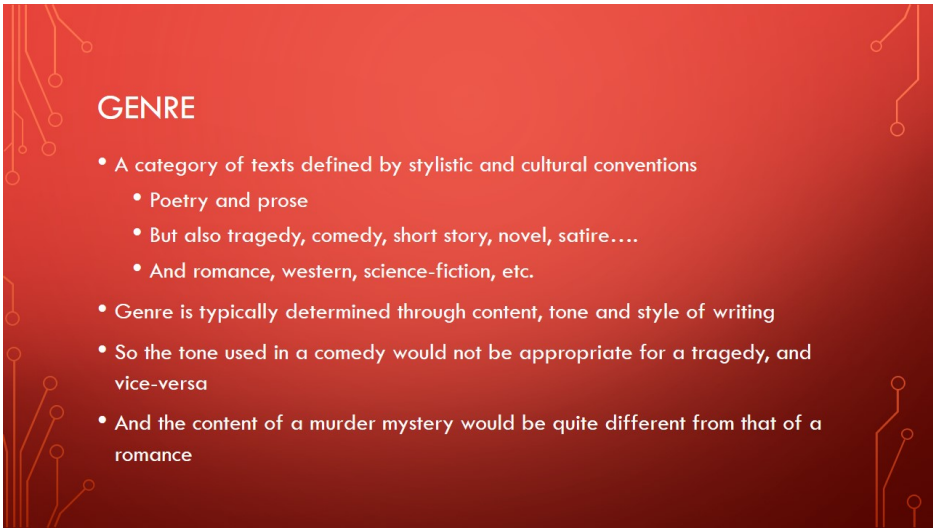


Slide 73

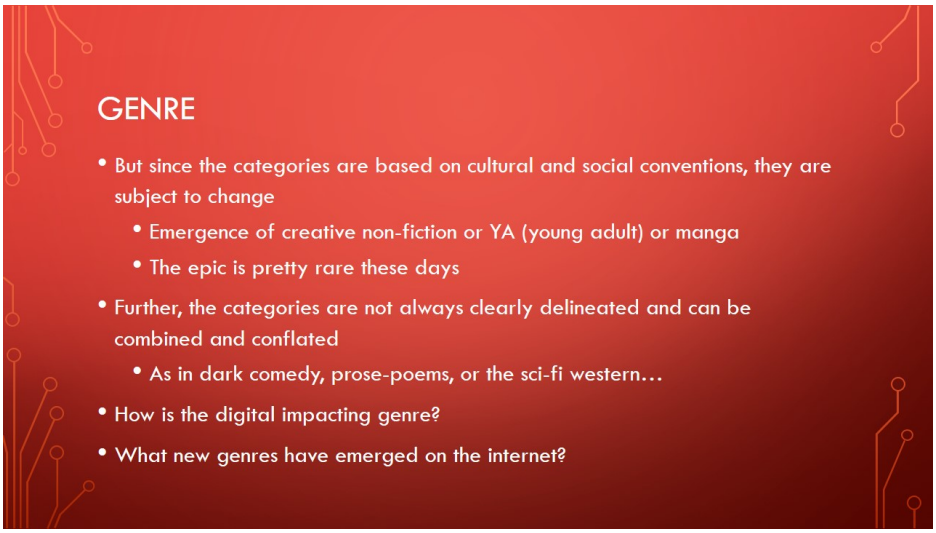
Chapter 5

Genre, Language, and Digital Documents

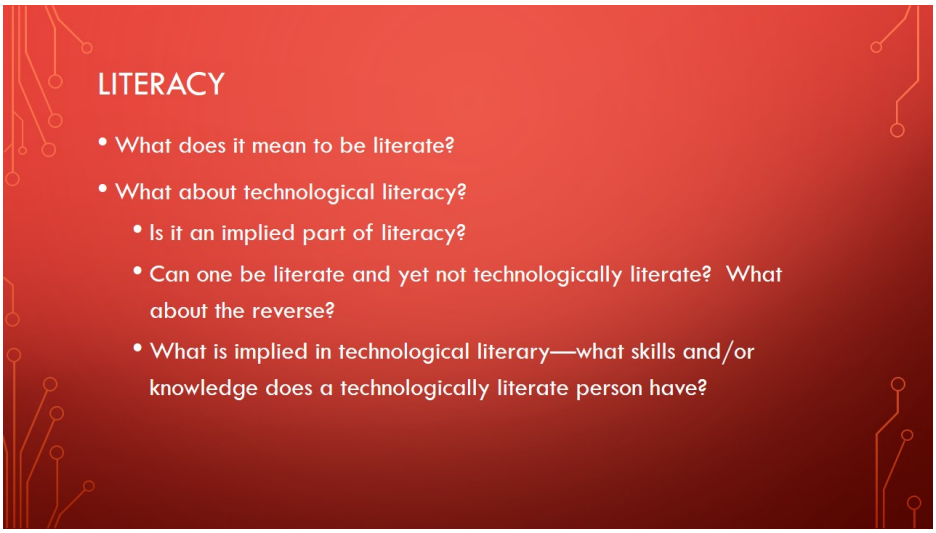
Slide 74



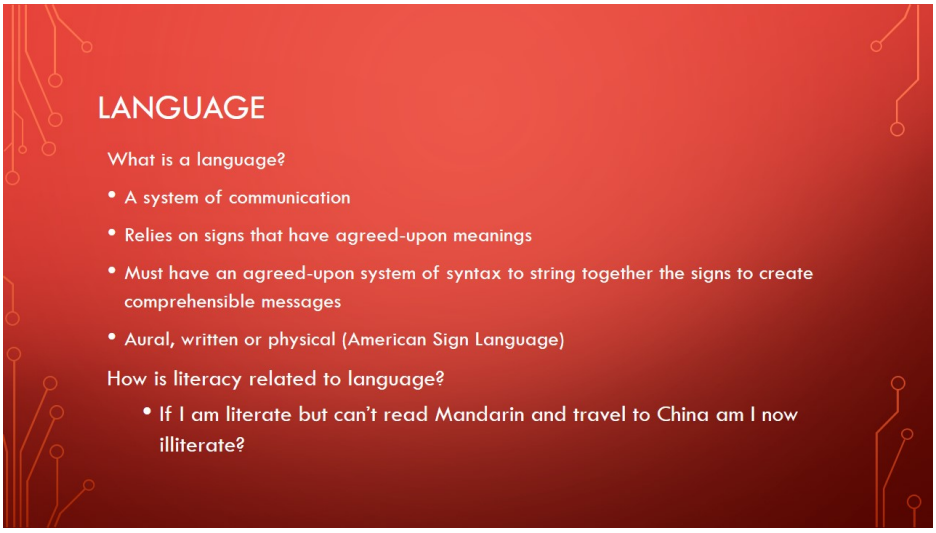
Slide 75



Slide 76



Slide 77



Slide 78

COMPUTER LANGUAGES

- Are computer languages actually languages?
- How are they different/similar to other languages?
- What is the purpose of computer (or programming languages)?
- When someone writes a computer program, "who" reads that message?
- How is that different from someone writing a letter to someone else?

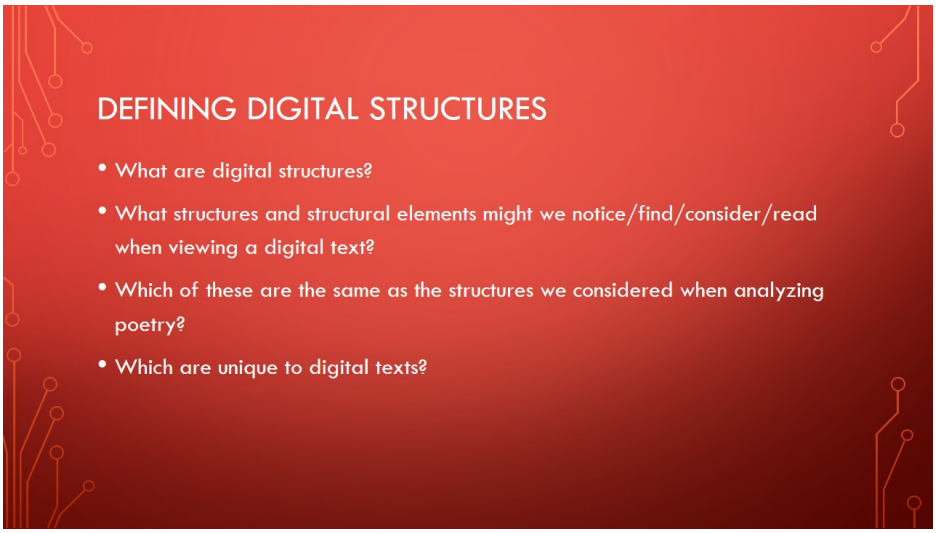
Slide 79

DEFINING DIGITAL TEXTS

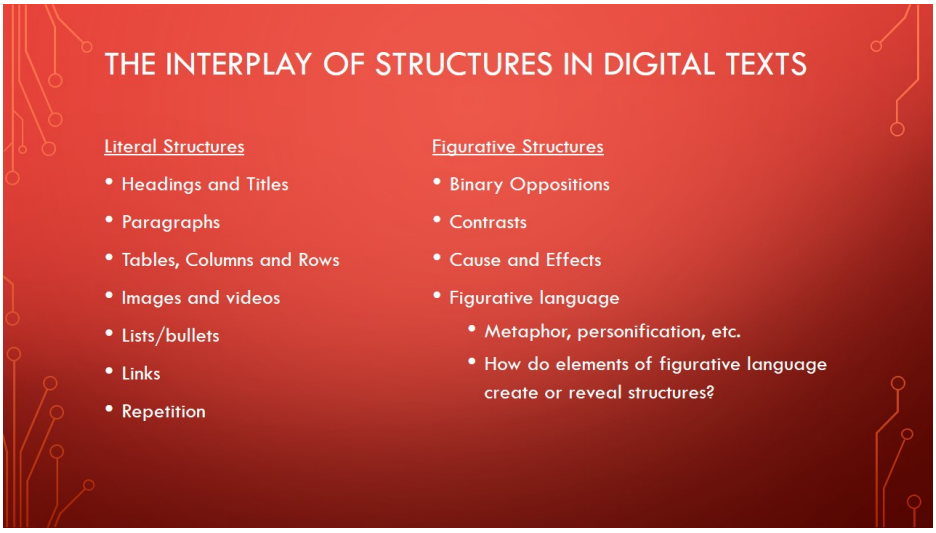
- What is a text?
- Is a text always a written document? (text=language?)
- What if we define text as simply: "that which can be read"?
- What does it mean to "read" something?
- Can we "read" an image? A video? A song? An advertisement?
- Can an image then be considered language? What does structuralism suggest about this question?
  - Word/image both = signifier

Slide 80

Slide 81



Slide 82



Slide 83

ANALYSIS OF A WEBPAGE

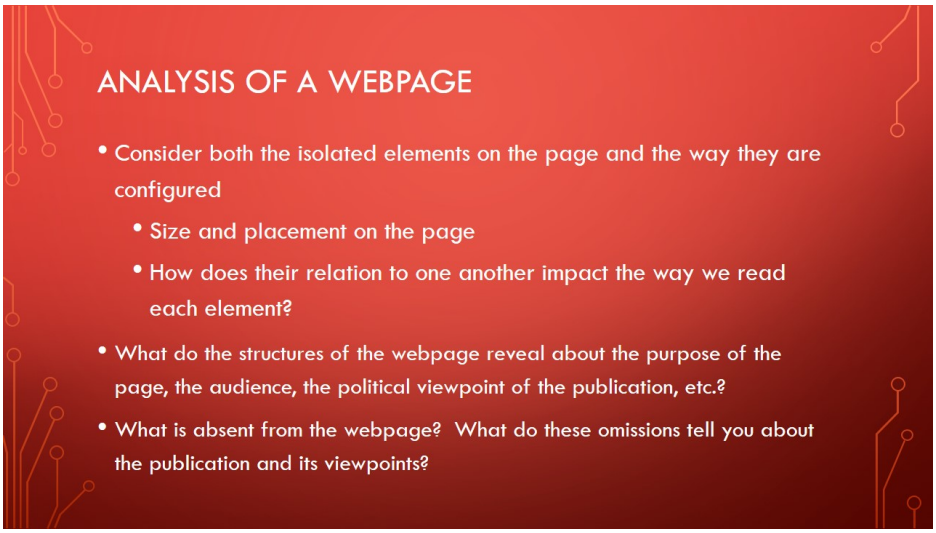
- How do you read this page?
- Literally how does your eye move over the page? Left to right? Up to down?
- What draws your attention and why?
- What hyperlinks would you follow?
- Remember what we know about how most people generally read webpages:
  - Reading horizontally
  - Reading only about 20 percent of the text on any given webpage
  - Browsing from page to page and/or following hyperlinks

Slide 84

ANALYSIS OF A WEBPAGE

- Map the webpage by looking for and identifying:
  - Literal structures
  - Figurative Structures
- What are the key terms on the page?
- What are the connotations of these key terms?
- Which terms or values are privileged?
- Which are disparaged?

Slide 85


ANALYSIS OF A WEBPAGE
• Consider both the isolated elements on the page and the way they are configured
• Size and placement on the page
• How does their relation to one another impact the way we read each element?
• What do the structures of the webpage reveal about the purpose of the page, the audience, the political viewpoint of the publication, etc.?
• What is absent from the webpage? What do these omissions tell you about the publication and its viewpoints?

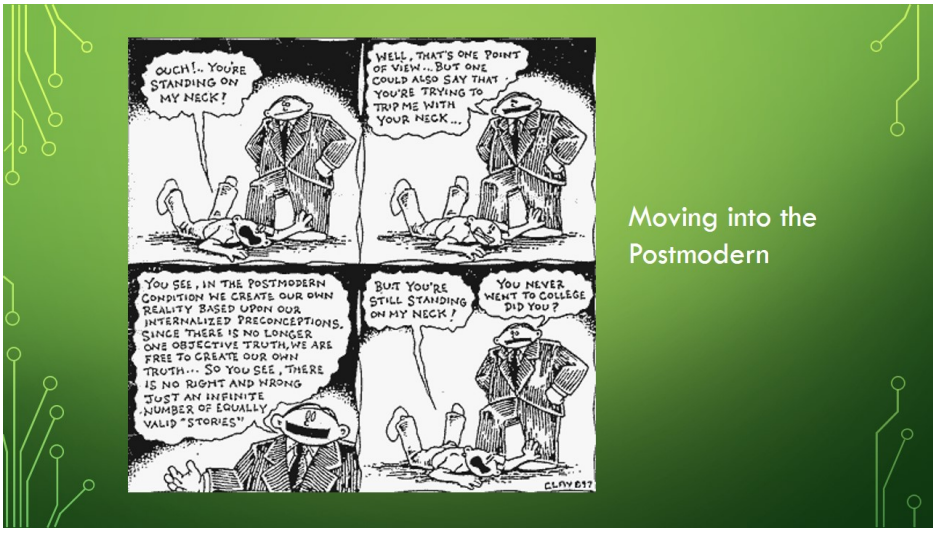
Slide 86

Chapter 6

Deconstruction

we start with the first stlide

Slide 87



Slide 88

MOVING INTO THE POSTMODERN

- There is no truth; truth is only and always subjective
- Truth is always a construct(ion)
- Truth is always relative so there are always and simultaneously many truths
- Since truth is always relative, there is no objective reality
- So all meaning is constructed through difference (there is no center)
- And that means meaning itself is always subjective and in flux

Slide 89

An optical illusion might also help illustrate the key ideas of postmodernism:

- Is this the image of an old woman or the image of a young woman?
- What determines how we view the image?

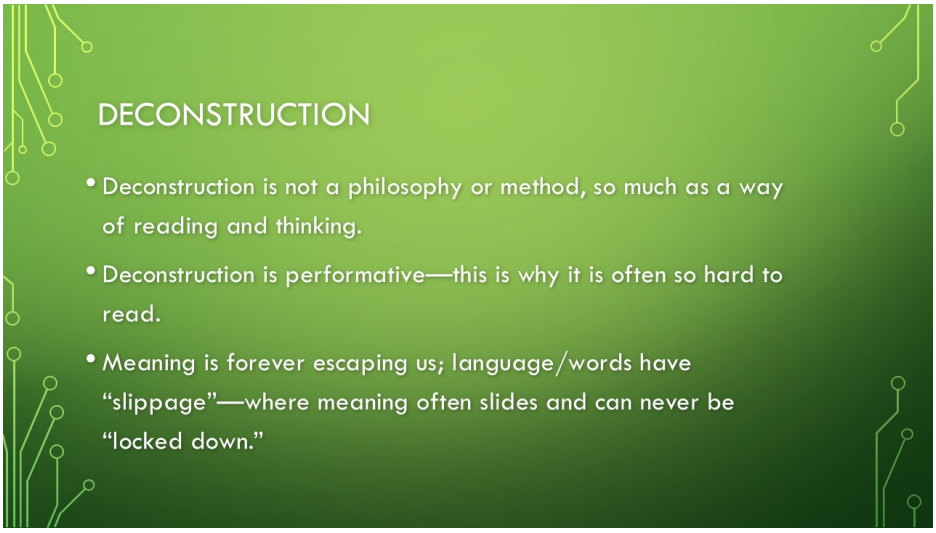
Slide 90

BUILDING FROM STRUCTURALISM

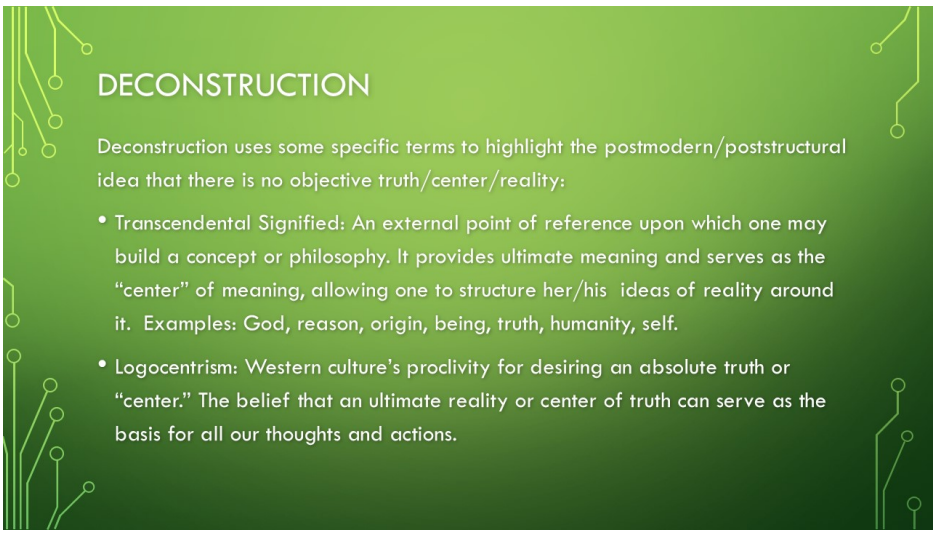
Derrida builds off of Saussure and structuralism in a few key areas:

- Meaning in language is constructed through the differences between/among signs
- Emphasis on intertextuality
- Binary oppositions are a fundamental structure within language and thought

Slide 91



Slide 92



Slide 93

DECONSTRUCTION: BINARY OPPOSITIONS

- Supplement: The unstable relationship between the two elements contained in a binary. Rather than being two distinct ideas/concepts, each informs the other. Each term helps define the other and is necessary for the other to exist.
- Différance: Two simultaneous meanings: 1) To defer, postpone, or delay; 2) to differ, to be different from. Caught in an endless chain of signifiers: each signifier differs from another, but only defers the other meaning.
- This is the essence of différance: meaning always is deferred and differing.

Slide 94

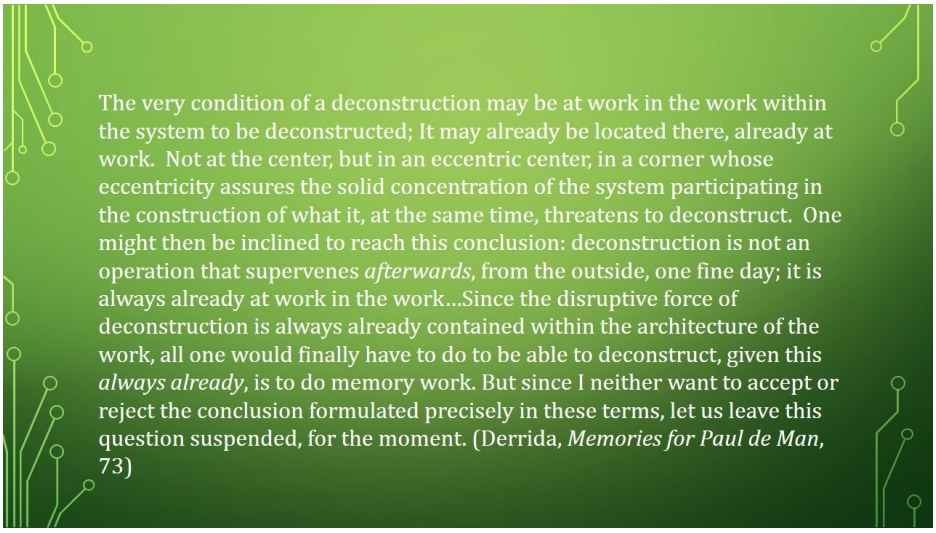
DECONSTRUCTION: BINARY OPPOSITIONS

- Deconstruction is interested in pulling apart, exploring and questions these binary oppositions
- Derrida does not believe we can get outside the binary (we are trapped in that deeply habitual way of thinking) but we can question the structure, question the binary itself by reversing the hierarchy
- Why would we do this? The goal is to gain new or more insight/knowledge into our own thinking and ways we create meaning(s)

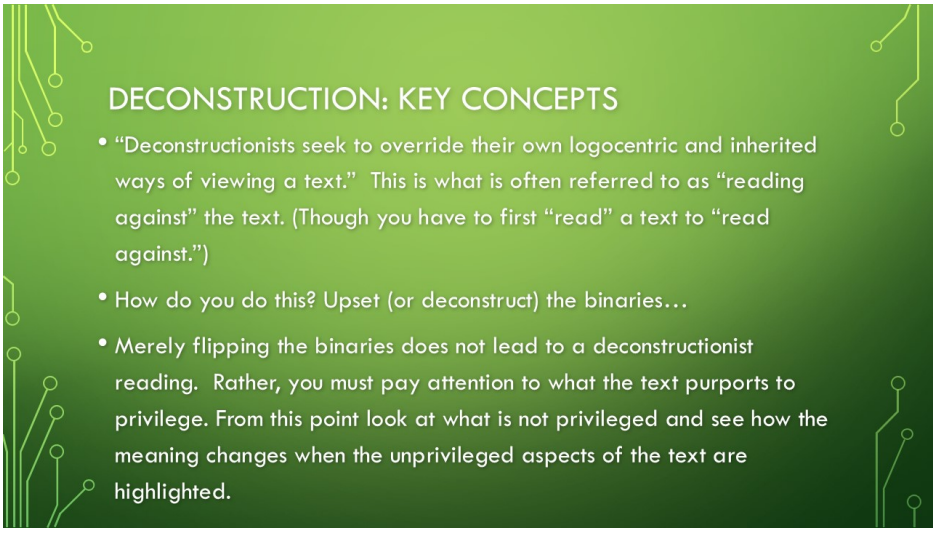
Slide 95

Derrida on Deconstruction:
https://www.youtube.com/watch?v=vgwOjjoYtco

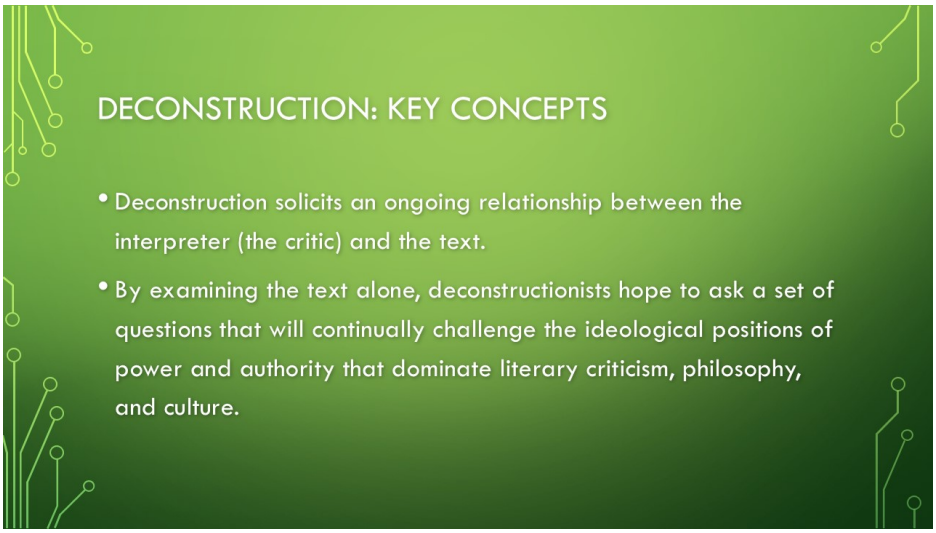
Slide 96



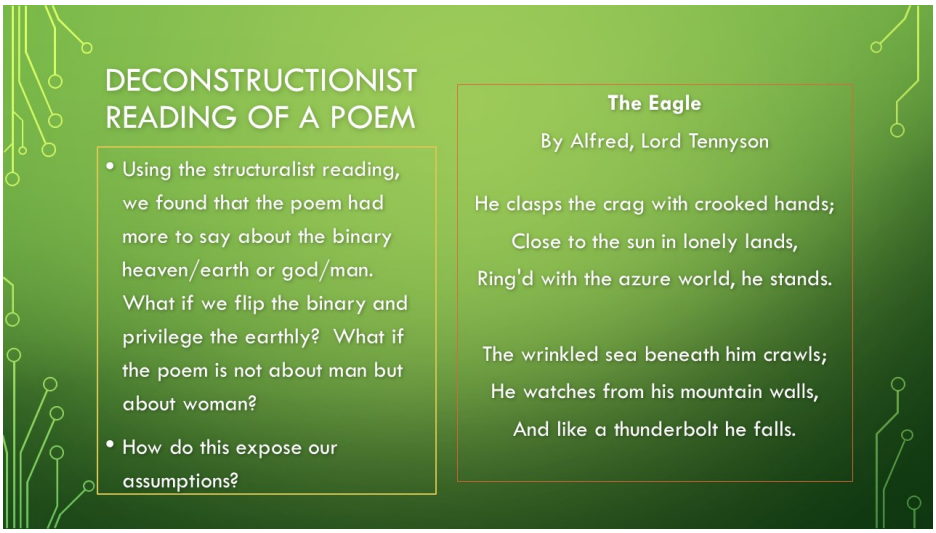
Slide 97



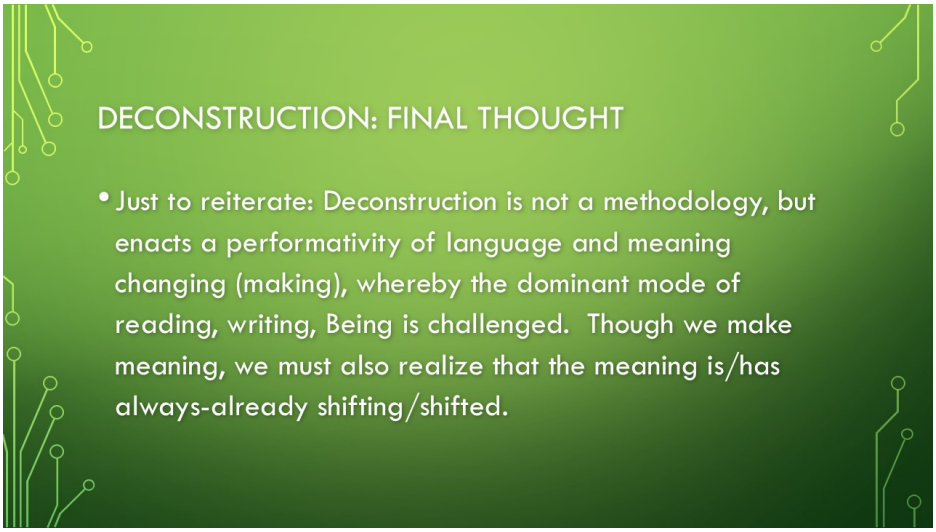
Slide 98



Slide 99



Slide 100



Slide 101

Chapter 7

Basic Concepts of the World Wide Web

The World Wide Web (WWWeb) is the hypertext/multimedia part of the Internet. It is implemented as a service on top of the Internet (at the aplication level) based on specific protocols and markup formats for documents.

Concepts of the World Wide Web

▷ **Definition 7.0.1** A web page is a document on the WWWeb that can include multimedia data and hyperlinks.

▷ **Definition 7.0.2** A web site is a collection of related Web pages usually designed or controlled by the same individual or company.

▷ a web site generally shares a common domain name.

▷ **Definition 7.0.3** A hyperlink is a reference to data that can immediately be followed by the user or that is followed automatically by a user agent.

▷ **Definition 7.0.4** A collection text documents with hyperlinks that point to text fragments within the collection is called a hypertext. The action of following hyperlinks in a hypertext is called browsing or navigating the hypertext.

▷ In this sense, the WWWeb is a multimedia hypertext.

7.1 Addressing on the World Wide Web

The essential idea is that the World Wide Web consists of a set of resources (documents, images, movies, etc.) that are connected by links (like a spider-web). In the WWWeb, the the links consist of pointers to addresses of resources. To realize them, we only need addresses of resources (much as we have IP numbers as addresses to hosts on the Internet).

Uniform Resource Identifier (URI), Plumbing of the Web

▷ **Definition 7.1.1** A uniform resource identifier (URI) is a global identifiers of network-retrievable documents (web resources). URIs adhere a uniform syntax (grammar) defined in RFC-3986 [BLFM05]. Grammar Rules contain: URI :== scheme, ′ :′, hierPart, [′?′ query], [′#′ fragment] hier − part :== ′//′ (pathAbempty | pathAbsolute | pathRootless

▷ **Example 7.1.2** The following are two example URIs and their component parts:

```
  http://example.com:8042/over/there?name=ferret#nose
  \__/   \______________/\_________/ \_________/ \__/
   |           |            |            |        |
scheme     authority       path        query   fragment
   |___   ____________________|_________
  /    \ /                             \
 mailto:m.kohlhase@jacobs-university.de
```

Note: URIs only identify documents, they do not have to be provide access to them (e.g. in a browser).

The definition above only specifies the structure of a URI and its functional parts. It is designed to cover and unify a lot of existing addressing schemes, including URLs (which we cover next), ISBN numbers (book identifiers), and mail addresses.

In many situations URIs still have to be entered by hand, so they can become quite unwieldy. Therefore there is a way to abbreviate them.

▷ Relative URIs

▷ **Definition 7.1.3** URIs can be abbreviated to relative URIs; missing parts are filled in from the context

▷ **Example 7.1.4** Relative URIs are more convenient to write

relative URI	abbreviates	in context
`#foo`	`⟪current-file⟫#foo`	curent file
`../bar.txt`	`file:///home/kohlhase/foo/bar.txt`	file system
`../bar.html`	`http://example.org/foo/bar.html`	on the web

Note that some forms of URIs can be used for actually locating (or accessing) the identified resources, e.g. for retrieval, if the resource is a document or sending to, if the resource is a mailbox. Such URIs are called "uniform resource *locators*", all others "uniform resource *locators*".

Uniform Resource Names and Locators

▷ **Definition 7.1.5** A uniform resource locator (URL) is a URI that that gives access to a web resource, by specifying an access method or location. All other URIs are called uniform resource names (URN).

▷ Idea: A URN defines the identity of a resource, a URL provides a method for finding it.

▷ **Example 7.1.6** The following URI is a URL (try it in your browser)

`http://kwarc.info/kohlhase/index.html`

▷ **Example 7.1.7** `urn:isbn:978-3-540-37897-6` only identifies [Koh06] (it is in the library)

▷ **Example 7.1.8** URNs can be turned into URL via a catalog service, e.g. `http://wm-urn.org/urn:isbn:978-3-540-37897-6`

▷ Note: URI/URLs are one of the core features of the web infrastructure, they are considered to be the plumbing of the WWWeb.(direct the flow of data)

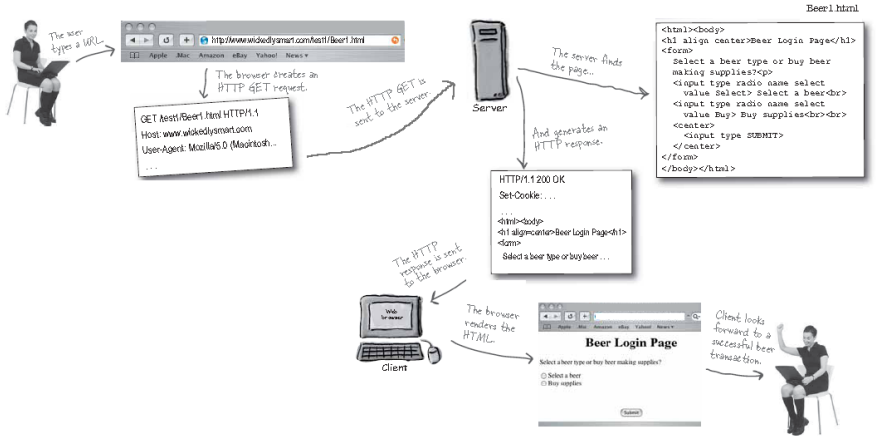
Historically, started out as URLs as short strings used for locating documents on the Internet. The generalization to identifiers (and the addition of URNs) as a concept only came about when the concepts evolved and the application layer of the Internet grew and needed more structure.

Note that there are two ways in URIs can fail to be resource locators: first, the scheme does not support direct access (as the ISBN scheme in our example), or the scheme specifies an access method, but address does not point to an actual resource that could be accessed. Of course, the problem of "dangling links" occurs everywhere we have addressing (and change), and so we will neglect it from our discussion. In practice, the URL/URN distinction is mainly driven by the scheme part of a URI, which specifies the access/identification scheme.

7.2 Running the World Wide Web

The infrastructure of the WWWeb relies on a client-server architecture, where the servers (called web servers) provide documents and the clients (usually web browsers) present the documents to the (human) users. Clients and servers communicate via the http protocol. We give an overview via a concrete example before we go into details.

The World Wide Web as a Client/Server System



We will now go through and introduce the infrastructure components of the WWWeb in the order we encounter them. We start with the user agent; in our example the web browser used by the user to request the web page by entering its URL into the URL bar.

Web Browsers

▷ **Definition 7.2.1** A web Browser is a software application for retrieving, presenting, and traversing information resources on the World Wide Web, enabling users to view Web pages and to jump from one page to another.

▷ Practical Browser Tools:

- ▷ Status Bar: security info, page load progress
- ▷ Favorites (bookmarks)
- ▷ View Source: view the code of a Web page
- ▷ Tools/Internet Options, history, temporary Internet files, home page, auto complete, security settings, programs, etc.

▷ **Example 7.2.2 (Common Browsers)**
- ▷ `MSInternetExplorer` is provided by Microsoft for `Windows` (very common)
- ▷ `FireFox` is an open source browser for all platforms, it is known for its standards compliance.
- ▷ `Safari` is provided by Apple for `MacOSX` and `Windows`
- ▷ `Chrome` is a lean and mean browser provided by Google
- ▷ `WebKit` is a library that forms the open source basis for `Safari` and `Chrome`.

The web browser communicates with the web server through a specialized protocol, the hypertext transfer protocol, which we cover now.

HTTP: Hypertext Transfer Protocol

▷ **Definition 7.2.3** The Hypertext Transfer Protocol (HTTP) is an application layer protocol for distributed, collaborative, hypermedia information systems.

▷ June 1999: HTTP/1.1 is defined in RFC 2616 [FGM+99].

Definition 7.2.4 HTTP is used by a client (called user agent) to access web resources (addressed by Uniform Resource Locators (URLs)) via a http request. The web server answers by supplying the resource

▷ Most important HTTP requests (5 more less prominent)

GET	Requests a representation of the specified resource.	safe
PUT	Uploads a representation of the specified resource.	idempotent
DELETE	Deletes the specified resource.	idempotent
POST	Submits data to be processed (e.g., from a web form) to the identified resource.	

▷ **Definition 7.2.5** We call a HTTP request safe, iff it does not change the state in the web server.(except for server logs, counters,. . . ; no side effects)

▷ **Definition 7.2.6** We call a HTTP request idempotent, iff executing it twice has the same effect as executing it once.

▷ HTTP is a stateless protocol (very memory-efficient for the server.)

Finally, we come to the last component, the web server, which is responsible for providing the web page requested by the user.

Web Servers

▷ **Definition 7.2.7** A web server is a network program that delivers web pages and supplementary resources to and receives content from user agents via the hypertext transfer protocol.

▷ **Example 7.2.8 (Common Web Servers)** ▷ apache is an open source web server that serves about 60% of the WWWeb.

▷ `IIS` is a proprietary server provided by Microsoft.

▷ `nginx` is a lightweight open source web server.

▷ Even though web servers are very complex software systems, they come pre-installed on most `UNIX` systems and can be downloaded for `Windows` [XAM].

Now that we have seen all the components we fortify our intuition of what actually goes down the net by tracing the http messages.

Example: An http request in real life

▷ Connect to the web server (port 80)(so that we can see what is happening)

```
telnet www.kwarc.info 80
```

▷ Send off the `GET` request

```
GET /teaching/GenCS2.html http/1.1
Host: www.kwarc.info
User-Agent: Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.6; en-US; rv:1.9.2.4)
  Gecko/20100413 Firefox/3.6.4
```

▷ Response from the server

```
HTTP/1.1 200 OK
Date: Mon, 03 May 2010 06:48:36 GMT
Server: Apache/2.2.9 (Debian) DAV/2 SVN/1.5.1 mod_fastcgi/2.4.6 PHP/5.2.6-1+lenny8 with
```

```
    Suhosin-Patch mod_python/3.3.1 Python/2.5.2 mod_ssl/2.2.9 OpenSSL/0.9.8g
Last-Modified: Sun, 02 May 2010 13:09:19 GMT
ETag: "1c78b-db1-4859c2f221dc0"
Accept-Ranges: bytes
Content-Length: 3505
Content-Type: text/html

<!--This file was generated by ws2html.xsl. Do NOT edit manually! -->
<html xmlns="http://www.w3.org/1999/xhtml"><head>...</head></html>
```


7.3 Multimedia Documents on the World Wide Web

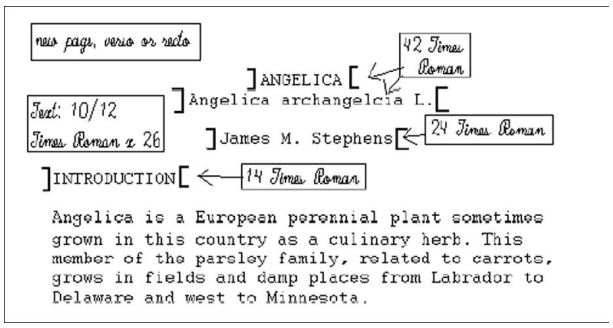
We have seen the client-server infrastructure of the WWWeb, which essentially specifies how hypertext documents are retrieved. Now we look into the documents themselves.

In Section3.0 have already discussed how texts can be encoded in files. But for the rich docments we see on the WWWeb, we have to realize that documents are more than just sequences of characters. This is traditionally captured in the notion of document markup.

Document Markup

▷ **Definition 7.3.1 (Document Markup)** Document markupmarkup is the process of adding codes (special, standardized character sequences) to a document to control the structure, formatting, or the relationship among its parts.

▷ **Example 7.3.2** A text with markup codes (for printing)



There are many systems for document markup ranging from informal ones as in Definition 7.3.1 that specify the intended document appearance to humans – in this case the printer – to technical ones which can be understood by machines but serving the same purpose.

WWWeb documents have a specialized markup language that mixes markup for document structure with layout markup, hyper-references, and interaction. The HTML markup elements always concern text fragments, they can be nested but may not otherwise overlap. This essentially turns a text into a document tree.

HTML: Hypertext Markup Language

▷ **Definition 7.3.3** The HyperText Markup Language (HTML), is a repre-

sentation format for web pages. Current version 4.01 is defined in [RHJ98].

▷ **Definition 7.3.4 (Main markup elements of HTML)** HTML marks up the structure and appearance of text with tags of the form `<el>` (begin) and `</el>` (end), where `el` is one of the following

structure	`html`,`head`, `body`	metadata	`title`, `link`, `meta`
headings	`h1`, `h2`, ..., `h6`	paragraphs	`p`, `br`
lists	`ul`, `ol`, `dl`, ..., `li`	hyperlinks	`a`
images	`img`	tables	`table`, `th`, `tr`, `td`, ...
styling	`style`, `div`, `span`	old style	`b`, `u`, `tt`, `i`, ...
interaction	`script`	forms	`form`, `input`, `button`

▷ **Example 7.3.5** A (very simple) HTML file with a single paragraph.

```
<html>
  <body>
    <p>Hello GenCS students!</p>
  </body>
</html>
```


HTML was created in 1990 and standardized in version 4 in 1997. Since then there has HTML has been basically stable, even though the WWWeb has evolved considerably from a web of static web pages to a Web in which highly dynamic web pages become user interfaces for web-based applications and even mobile applets. Acknowledging the growing discrepancy, the W3C has started the standardization of version 5 of HTML.

HTML5: The Next Generation HTML

▷ **Definition 7.3.6** The HyperText Markup Language (HTML5), is believed to be the next generation of HTML. It is defined by the W3C and the WhatWG.

▷ HTML5 includes support for

- ▷ audio/video without plugins,
- ▷ a canvas element for scriptable, 2D, bitmapped graphics
- ▷ SVG for Scalable Vector Graphics
- ▷ MathML inline and display-style mathematical formulae

▷ The W3C is expected to issue a "recommendation" that standardizes HTML5 in 2014.

▷ Even though HTML5 is not formally standardized yet, almost all major web browsers already implement almost all of HTML5.

As the WWWeb evolved from a hypertext system purely aimed at human readers to an Web of multimedia documents, where machines perform added-value services like searching or aggregating, it became more important that machines could understand critical aspects web pages. One way

to facilitate this is to separate markup that specifies the content and functionality from markup that specifies human-oriented layout and presentation (together called "styling"). This is what "cascading style sheets" set out to do. Another motivation for CSS is that we often want the styling of a web page to be customizable (e.g. for vision-impaired readers).

CSS: Cascading Style Sheets

▷ Idea: Separate structure/function from appearance.

Definition 7.3.7 The Cascading Style Sheets (CSS), is a style sheet language that allows authors and users to attach style (e.g., fonts and spacing) to structured documents. Current version 2.1 is defined in [BCHL09].

▷ **Example 7.3.8** Our text file from Example 7.3.5 with embedded CSS

```
<html>
  <head>
    <style type="text/css">
       body {background-color:#d0e4fe;}
       h1 {color:orange;
                text-align:center;}
       p {font-family:"Verdana";
                font-size:20px;}
    </style>
  </head>
  <body>
    <h1>CSS example</h1>
    <p>Hello GenCSII!.</p>
  </body>
</html>
```

CSS example

Hello GenCSII!.

Chapter 8

Web Applications

In this chapter we show how with a few additions to the basic WWWeb infrastructure introduced in Chapter6, we can turn web pages into web-based applications that can be used without having to install additional software.

The first thing we need is a means to send information back to the web server, which can be used as input for the web application. Fortunately, this is already forseen by the HTML format.

HTML Forms: Submitting Information to the Web Server

▷ **Example 8.0.9** Forms contain input fields and explanations.

```
<form name="input" action="html_form_submit.asp" method="get">
  Username: <input type="text" name="user" />
  <input type="submit" value="Submit" />
</form>
```

The result is a form with three elements: a text, an input field, and a submit button, that will trigger a HTTP GET request to the URL specified in the `action` attribute.

Username: Submit

As the WWWeb is based on a client-server architecture, computation in web applications can be executed either on the client (the web browser) or the server (the web server). For both we have a special technology; we start with computation on the web server.

Server-Side Scripting: Programming Web Pages

▷ Idea: Why write HTML pages if we can also program them! (easy to do)

▷ **Definition 8.0.10** A server-side scripting framework is a web server extension that generates web pages upon HTTP GET requests.

▷ **Example 8.0.11** `perl` is a scripting language with good string manipulation facilities. `perl` CGI is an early server-side scripting framework based on this.

▷ Server-side scripting frameworks allow to make use of external resources (e.g. databases or data feeds) and computational services during web page generation.

▷ Problem: Most web page content is static (page head, text blocks, etc.) (and no HTML editing support in program editors)

▷ Idea: Embed program snippets into HTML pages.(only execute these, copy rest)

▷ **Definition 8.0.12** A server-side scripting language is a server side scripting framework where web pages are generated from HTML documents with embedded program fragments that are executed in context during web page generation.

▷ Note: No program code is left in the resulting web page after generation (important security concern)

To get a concrete intuition on the possibilities of server-side scripting frameworks, we will present PHP, a commonly used open source scripting framework. There are many other examples, but they mainly differ on syntax and advanced features.

PHP, a Server-Side Scripting Language

▷ **Definition 8.0.13** PHP (originally "Programmable Home Page Tools", later "PHP: Hypertext Processor") is a server-side scripting language with a C-like syntax. PHP code is embedded into HTML via special "tags" <?php and ?>

▷ **Example 8.0.14** The following PHP program uses echo for string output

```
<html>
  <body><?php echo 'Hello world';?></body>
</html>
```

▷ **Example 8.0.15** We can access the server clock in PHP (and manipulate it)

```
<?php
$tomorrow = mktime(0,0,0,date("m"),date("d")+1,date("Y"));
echo "Tomorrow is ".date("d. m. Y", $tomorrow);
?>
This fragment inserts tomorrow's date into a web page
```

▷ **Example 8.0.16** We can generate pages from a database (here MySQL)

```
<?php
$con = mysql_connect("localhost","peter","abc123");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }

mysql_select_db("my_db", $con);

$result = mysql_query("SELECT * FROM Persons");
```

```
while($row = mysql_fetch_array($result))
  {
  echo $row['FirstName'] . " " . $row['LastName'];
  echo "<br />";
  }

mysql_close($con);
?>
```

▷ **Example 8.0.17** We can even send e-mail via this e-mail form.

```
<html><body>
<?php
if (isset($_REQUEST['email']))//if "email" is filled out, send email
  {//send email
  $email = $_REQUEST['email'] ;
  $subject = $_REQUEST['subject'] ;
  $message = $_REQUEST['message'] ;
  mail("someone@example.com", $subject,
  $message, "From:" . $email);
  echo "Thank you for using our mail form";}
else //if "email" is not filled out, display the form
  {echo "<form method='post' action='mailform.php'>
   Email: <input name='email' type='text' /><br />
   Subject: <input name='subject' type='text' /><br />
   Message:<br />
   <textarea name='message' rows='15' cols='40'>
   </textarea><br />
   <input type='submit' />
   </form>";}
?>
</body></html>
```


With server-side scripting frameworks like PHP, we can already build web applications, which we now define.

Web Applications: Using Applications without Installing

▷ **Definition 8.0.18** A web application is a website that serves as a user interface for a server-based application using a web browser as the client.

▷ **Example 8.0.19** Commonly used web applications include

- ▷ `http://ebay.com`; auction pages are generated from databases
- ▷ `http://www.weather.com`; weather information generated weather feeds
- ▷ `http://slashdot.org`; aggregation of news feeds/discussions
- ▷ `http://github.com`; source code hosting and project management

Common Traits: pages generated from databases and external feeds, content submission via HTML forms, file upload

▷▷ **Definition 8.0.20** A web application framework is a software framework for creating web applications.

▷ **Example 8.0.21** The LAMP stack is a web application framework based on `linux`, `apache`, `MySQL`, and PHP.

▷ **Example 8.0.22** A variant of the LAMP stack is available for `Windows` as XAMPP [XAM].

Indeed, the first web applications were essentially built in this way. Note however, that as we remarked above, no PHP code remains in the generated web pages, which thus "look like" static web pages to the client, even though they were generated dynamically on the server.

There is one problem however with web applications that is difficult to solve with the technologies so far. We want web applications to give the user a consistent user experience even though they are made up of multiple web pages. In a regular application we we only want to login once and expect the application to remember e.g. our username and password over the course of the various interactions with the system. For web applications this poses a technical problem which we now discuss.

State in Web Applications and Cookies

▷ Recall: Web applications contain multiple pages, HTTP is a stateless protocol.

▷ Problem: how do we pass state between pages?(e.g. username, password)

▷ Simple Solution: Pass information along in query part of page URLs.

▷ **Example 8.0.23 (HTTP GET for Single Login)** Since we are generating pages we can generated augmented links

```
<a href="http://example.org/more.html?user=joe,pass=hideme">... more</a>
```

Problem: only works for limited amounts of information and for a single session

▷▷ Other Solution: Store state persistently on the client hard disk

▷ **Definition 8.0.24** A cookie is a text file stored on the client hard disk by the web browser. Web servers can request the browser to store and send cookies.

▷ cookies are data not programs, they do not generate pop-ups or behave like viruses, but they can include your log-in name and browser preferences

▷ cookies can be convenient, but they can be used to gather information about you and your browsing habits

▷ **Definition 8.0.25** third party cookies are used by advertising companies to track users across multiple sites.(but you can turn off, and even delete cookies)

Note that that both solutions to the state problem are not ideal, for usernames and passwords the URL-based solution is particularly problematic, since HTTP transmits URLs in GET requests without encryption, and in our example passwords would be visible to anybody with a packet

sniffer. Here cookies are little better as cookies, since they can be requested by any website you visit.

We now turn to client-side computation

One of the main advantages of moving documents from their traditional ink-on-paper form into an electronic form is that we can interact with them more directly. But there are many more interactions than just browsing hyperlinks we can think of: adding margin notes, looking up definitions or translations of particular words, or copy-and-pasting mathematical formulae into a computer algebra system. All of them (and many more) can be made, if we make documents programmable. For that we need three ingredients: *i)* a machine-accessible representation of the document structure, and *ii)* a program interpreter in the web browser, and *iii)* a way to send programs to the browser together with the documents. We will sketch the WWWeb solution to this in the following.

Dynamic HTML

▷ Observation: The nested, markup codes turn HTML documents into trees.

▷ **Definition 8.0.26** The document object model (DOM) is a data structure for the HTML document tree together with a standardized set of access methods.

▷ Note: All browsers implement the DOM and parse HTML documents into it; only then is the DOM rendered for the user.

▷ Idea: generate parts of the web page dynamically by manipulating the DOM.

▷ **Definition 8.0.27** JavaScript is an object-oriented scripting language mostly used to enable programmatic access to the DOM in a web browser.

▷ JavaScript is standardized by ECMA in [ECM09].

▷ **Example 8.0.28** We write the some text into a HTML document object (the document API)

```
<html>
 <head>
  <script type="text/javascript">document.write("Dynamic HTML!");</script>
 </head>
 <body><!-- nothing here; will be added by the script later --></body>
</html>
```


Let us fortify our intuition about dynamic HTML by going into a more involved example.

Applications and useful tricks in Dynamic HTML

▷ **Example 8.0.29** hide document parts by setting CSS `style` attribs to `display:none`

```
<html>
  <head>
    <style type="text/css">#dropper { display: none; }</style>
    <script language="JavaScript" type="text/javascript">
      function toggleDiv(element){
          if(document.getElementById(element).style.display == 'none')
```

```
            {document.getElementById(element).style.display = 'block'}
        else if(document.getElementById(element).style.display == 'block')
            {document.getElementById(element).style.display = 'none'}}
    </script>
  </head>
  <body>
    <div onClick="toggleDiv('dropper');">...more </div>
    <div id="dropper">
      <p>Now you see it!</p>
    </div>
  </body>
</html>
```

Application: write "gmail" or "google docs" as JavaScript enhanced web applications. (client-side computation for immediate reaction)

⋙ Current Megatrend: Computation in the "cloud", browsers (or "apps") as user interfaces

Current web applications include simple office software (word processors, online spreadsheets, and presentation tools), but can also include more advanced applications such as project management, computer-aided design, video editing and point-of-sale. These are only possible if we carefully balance the effects of server-side and client-side computation. The former is needed for computational resources and data persistence (data can be stored on the server) and the latter to keep personal information near the user and react to local context (e.g. screen size).

Chapter 9

An Overview over XML Technologies

Excursion: XML (EXtensible Markup Language)

- ▷ XML is language family for the Web
  - ▷ tree representation language (begin/end brackets)
  - ▷ restrict instances by *Doc. Type Def. (DTD)* or *Schema* (Grammar)
  - ▷ Presentation markup by *style files* (XSL: XML Style Language)
- ▷ XML is extensible HTML & simplified SGML
- ▷ logic annotation (*markup*) instead of presentation!
- ▷ many tools available: parsers, compression, data bases, ...
- ▷ conceptually: transfer of directed graphs instead of strings.
- ▷ details at `http://www.w3c.org`

The idea of XML being an "extensible" markup language may be a bit of a misnomer. It is made "extensible" by giving language designers ways of specifying their own vocabularies. As such XML does not have a vocabulary of its own, so we could have also it an "empty" markup language that can be filled with a vocabulary.

XML is Everywhere (E.g. document metadata)

- ▷ **Example 9.0.30** Open a `PDF` file in `AcrobatReader`, then cklick on $File \searrow DocumentProperties \searrow DocumentMetadata \searrow ViewSource$, you get the following text: (showing only a small part)

```
<rdf:RDF xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
         xmlns:iX='http://ns.adobe.com/iX/1.0/'>
  <rdf:Description xmlns:pdf='http://ns.adobe.com/pdf/1.3/'>
    <pdf:CreationDate>2004-09-08T16:14:07Z</pdf:CreationDate>
    <pdf:ModDate>2004-09-08T16:14:07Z</pdf:ModDate>
    <pdf:Producer>Acrobat Distiller 5.0 (Windows)</pdf:Producer>
```

```
    <pdf:Author>Herbert Jaeger</pdf:Author>
    <pdf:Creator>Acrobat PDFMaker 5.0 for Word</pdf:Creator>
    <pdf:Title>Exercises for ACS 1, Fall 2003</pdf:Title>
  </rdf:Description>
  ...
  <rdf:Description xmlns:dc='http://purl.org/dc/elements/1.1/'>
    <dc:creator>Herbert Jaeger</dc:creator>
    <dc:title>Exercises for ACS 1, Fall 2003</dc:title>
  </rdf:Description>
</rdf:RDF>
```


This is an excerpt from the document metadata which `AcrobatDistiller` saves along with each `PDF` document it creates. It contains various kinds of information about the creator of the document, its title, the software version used in creating it and much more. Document metadata is useful for libraries, bookselling companies, all kind of text databases, book search engines, and generally all institutions or persons or programs that wish to get an overview of some set of books, documents, texts. The important thing about this document metadata text is that it is not written in an arbitrary, `PDF`-proprietary format. Document metadata only make sense if these metadata are independent of the specific format of the text. The metadata that `MSWord` saves with each Word document should be in the same format as the metadata that Amazon saves with each of its book records, and again the same that the British library uses, etc.

XML is Everywhere (E.g. Web Pages)

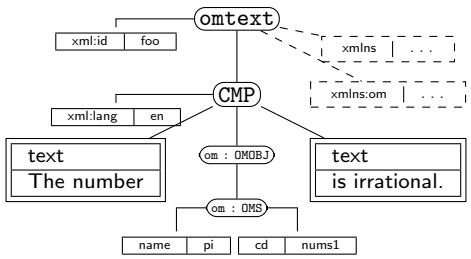
▷ **Example 9.0.31** Open web page file in `FireFox`, then click on $View \searrow PageSource$, you get the following text: (showing only a small part and reformatting)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
                      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Michael Kohlhase</title>
    <meta name="generator"
          content="Page generated from XML sources with the WSML package"/>
  </head>
  <body>...
    <p>
      <i>Professor of Computer Science</i><br/>
      Jacobs University<br/><br/>
      <strong>Mailing address - Jacobs (except Thursdays)</strong><br/>
      <a href="http://www.jacobs-university.de/schools/ses">
        School of Engineering &amp;amp; Science
       </a><br/>...
    </p>...
  </body>
</html>
```


XML Documents as Trees

▷ Idea: An XML Document is a Tree

```
<omtext xml:id="foo"
   xmlns="..."
   xmlns:om="...">
  <CMP xml:lang='en'>
   The number
   <om:OMOBJ>
     <om:OMS cd="nums1"
             name="pi"/>
   <om:OMOBJ>
   is irrational.
  </CMP>
</omtext>
```



▷ **Definition 9.0.32** The XML document tree is made up of element nodes, attribute nodes, text nodes (and namespace declarations, comments,...)

▷ **Definition 9.0.33** For communication this tree is serialized into a balanced bracketing structure, where

- ▷ an element `el` is represented by the brackets $<$ `el` $>$ (called the opening tag) and $<$ /`el` $>$ (called the closing tag).
- ▷ The leaves of the tree are represented by empty elements (serialized as $<$ `el` $><$ /`el` $>$, which can be abbreviated as $<$ `el`/ $>$
- ▷ and text nodes (serialized as a sequence of UniCode characters).
- ▷ An element node can be annotated by further information using attribute nodes — serialized as an attribute in its opening tag

Note: As a document is a tree, the XML specification mandates that there must be a unique document root.

▷ Internet Standardization

▷ Question: Where do all the protocols come from?(someone has to manage that)

▷ **Definition 9.0.34** The Internet Engineering Task Force (IETF) is an open standards organization that develops and standardizes Internet standards, in particular the TCP/IP and Internet protocol suite.

▷ All participants in the IETF are volunteers(usually paid by their employers)

▷ Rough Consensus and Running Code: Standards are determined by the "rough consensus method" (consensus preferred, but not all members need agree) IETF is interested in practical, working systems that can be quickly implemented.

▷ Idea: running code leads to rough consensus or vice versa.

▷ **Definition 9.0.35** The standards documents of the IETF are called Request for Comments (RFC). (more than 6300 so far; see `http://www.rfc-editor.org/`)

The Dual Role of Grammar in XML (I)

▷ The XML specification [XML] contains a large character-level grammar. (81 productions)

NameChar :== Letter | Digit | '.' | '−' | '_' | ':' | CombiningChar | Extender

Name :== (Letter | '_' | ':') (NameChar)*

element :== EmptyElementTag | STag content ETag

STag :== '<' $(S)^*$ Name $(S)^*$ attribute $(S)^*$ '>'

ETag :== '< /' $(S)^*$ Name $(S)^*$ '>'

EmptyElementTag :== '<' $(S)^*$ Name $(S)^*$ attribute $(S)^*$ '/ >'

▷ use these to parse well-formed XML document into a tree data structure

▷ use these to serialize a tree data structure into a well-formed XML document

▷ Idea: Integrate XML parsers/serializers into all programming languages to communicate trees instead of strings. (more structure $\hat{=}$ better CS)

The Dual Role of Grammar in XML (II)

▷ Idea: We can define our own XML language by defining our own elements and attributes.

▷ Validation: Specify your language with a tree grammar(works like a charm)

▷ **Definition 9.0.36** Document Type Definitions (DTDs) are grammars that are built into the XML framework.

Put <DOCTYPE foo PUBLIC "foo.dtd"¿! into the second line of the document to validate.

▷ **Definition 9.0.37** RelaxNG is a modern XML grammar/schema framework on top of the XML framework.

RelaxNG, A tree Grammar for XML

▷ **Definition 9.0.38** RelaxNG (RelaxNG: Regular Language for XML Next Generation) is a tree grammar framework for XML documents.

A RelaxNG schema is itself an XML document; however, RelaxNG also offers a popular, non-XML compact syntax.

▷ **Example 9.0.39** The RelaxNG grammars validate the left document

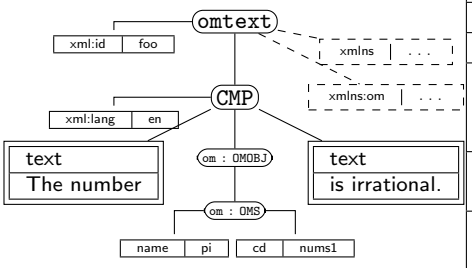
document	RelaxNG in XML
<lecture> <slide id="foo"> first slide </slide> <slide id="bar"> second one </slide> </lecture>	<grammar> <start> <element name="le <oneOrMore> <ref name="sli </oneOrMore> </element> </start> <define name="sli <element name=" <text/> </element> <attribute name <text/> </attribute> </define> </grammar>

One of the great advantages of viewing marked-up documents as trees is that we can describe subsets of its nodes.

XPath, A Language for talking about XML Tree Fragments

▷ **Definition 9.0.40** The XML path language (XPath) is a language framework for specifying fragments of XML trees.

▷ **Example 9.0.41**



XPath exp.	fragment
/	root
omtext/CMP/*	all CMP children
//@name	the name attribute on the om : OMS element
//CMP/*[1]	the first child of all OMS elements
//*[@cd='nums1']	all elements whose cd has value nums1

An XPath processor is an application or library that reads an XML file into a DOM and given an XPath expression returns (pointers to) the set of nodes in the DOM that satisfy the expression.

XSLT, A tree Transformer for XML

▷ **Definition 9.0.42** XSLT (Extensible Stylesheet Language Transformations) is a declarative, XML-based language used for the transformation of XML documents. It is standardized by the W3C.

▷ **Definition 9.0.43** XSLT stylesheets consist of a set of templates which match a XML elements via an XPath expression and create a result tree.

▷ **Definition 9.0.44** An XSLT Processor is a program that takes an XSLT stylesheet S and an XML file X as input and transforms X as specified by the templates in S.

▷ **Example 9.0.45** There are various open source or free XSLT processors

- ▷ `xsltproc` [Vei] is very fast, but only supports XSLT version 1.
- ▷ `saxon` [Kay08] supports XSLT version 2, but is slower.

▷ **Example 9.0.46** Use this stylesheet to extract a numbered table of contents from an HTML document

```
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:template match="/">
   <html><body><xsl:apply-templates select="//h1"/></body></html>
 </xsl:template>

 <xsl:template match="*"/>

 <xsl:template match="h1">
   <p style="font-size:large">
     <xsl:value-of select="preceeding-sibling::h1"/>
     <xsl:copy-of select="*|text()"/>
   </p>
 </xsl:template>
</xsl:stylesheet>
```

Part II

Mechanics and Consequences of Digital Media

In this part of the course we introduce the mechanics of digital media – how do we create, transform, and manage digital documetns, and discuss how digical media affect individuals and society[8] EdN:8

[8] EdNote: MK: continue

Chapter 10

Legal Foundations of Information Technology

In this chapter, we cover a topic that is a very important secondary aspect of our work as Computer Scientists: the legal foundations that regulate how the fruits of our labor are appreciated (and recompensated), and what we have to do to respect people's personal data.

10.1 Intellectual Property, Copyright, and Licensing

The first complex of questions centers around the assessment of the products of work of knowledge/information workers, which are largely intangible, and about questions of recompensation for such work.

Intellectual Property: Concept

▷ Question: Intellectual labour creates (intangible) objects, can they be owned?

▷ Answer: Yes: in certain circumstances they are property like tangible objects.

▷ **Definition 10.1.1** The concept of intellectual property motivates a set of laws that regulate property rights on intangible objects, in particular

- ▷ Patents grant exploitation rights on original ideas.
- ▷ Copyrights grant personal and exploitation rights on expressions of ideas.
- ▷ Industrial Design Rights protect the visual design of objects beyond their function.
- ▷ Trademarks protect the signs that identify a legal entity or its products to establish brand recognition.

▷ Intent: Property-like treatment of intangibles will foster innovation by giving individuals and organizations material incentives.

Naturally, many of the concepts are hotly debated. Especially due to the fact that intuitions and legal systems about property have evolved around the more tangible forms of properties that cannot be simply duplicated and indeed multiplied by copying them. In particular, other intangibles like physical laws or mathematical theorems cannot be property.

Intellectual Property: Problems

- ▷ Delineation Problems: How can we distinguish the product of human work, from "discoveries", of e.g. algorithms, facts, genome, algorithms. (not property)
- ▷ Philosophical Problems: The implied analogy with physical property (like land or an automobile) fails because physical property is generally rivalrous while intellectual works are non-rivalrous (the enjoyment of the copy does not prevent enjoyment of the original).
- ▷ Practical Problems: There is widespread criticism of the concept of intellectual property in general and the respective laws in particular.
  - ▷ (software) patents are often used to stifle innovation in practice. (patent trolls)
  - ▷ copyright is seen to help big corporations and to hurt the innovating individuals

We will not go into the philosophical debates around intellectual property here, but concentrate on the legal foundations that are in force now and regulate IP issues. We will see that groups holding alternative views of intellectual properties have learned to use current IP laws to their advantage and have built systems and even whole sections of the software economy on this basis.

Many of the concepts we will discuss here are regulated by laws, which are (ultimately) subject to national legislative and juridicative systems. Therefore, none of them can be discussed without an understanding of the different jurisdictions. Of course, we cannot go into particulars here, therefore we will make use of the classification of jurisdictions into two large legal traditions to get an overview. For any concrete decisions, the details of the particular jurisdiction have to be checked.

Legal Traditions

- ▷ The various legal systems of the world can be grouped into "traditions".
- ▷ **Definition 10.1.2** Legal systems in the common law tradition are usually based on case law, they are often derived from the British system.
- ▷ **Definition 10.1.3** Legal systems in the civil law tradition are usually based on explicitly codified laws (civil codes).
- ▷ As a rule of thumb all English-speaking countries have systems in the common law tradition, whereas the rest of the world follows a civil law tradition.

Another prerequisite for understanding intellectual property concepts is the historical development

of the legal frameworks and the practice how intellectual property law is synchronized internationally.

Historic/International Aspects of Intellectual Property Law

▷ Early History: In late antiquity and the middle ages IP matters were regulated by royal privileges

▷ History of Patent Laws: First in Venice 1474, Statutes of Monopolies in England 1624, US/France 1790/1...

▷ History of Copyright Laws: Statue of Anne 1762, France: 1793, ...

▷ Problem: In an increasingly globalized world, national IP laws are not enough.

▷ **Definition 10.1.4** The Berne convention process is a series of international treaties that try to harmonize international IP laws. It started with the original Berne convention 1886 and went through revision in 1896, 1908, 1914, 1928, 1948, 1967, 1971, and 1979.

▷ The World Intellectual Property Organization Copyright Treaty was adopted in 1996 to address the issues raised by information technology and the Internet, which were not addressed by the Berne Convention.

▷ **Definition 10.1.5** The Anti-Counterfeiting Trade Agreement (ACTA) is a multinational treaty on international standards for intellectual property rights enforcement.

▷ With its focus on enforcement ACTA is seen my many to break fundamental human information rights, criminalize FLOSS

10.1.1 Copyright

In this subsection, we go into more detail about a central concept of intellectual property law: copyright is the component most of IP law applicable to the individual computer scientist. Therefore a basic understanding should be part of any CS education. We start with a definition of what works can be copyrighted, and then progress to the rights this affords to the copyright holder.

Copyrightable Works

▷ **Definition 10.1.6** A copyrightable work is any artefact of human labor that fits into one of the following eight categories:

- ▹ Literary works: Any work expressed in letters, numbers, or symbols, regardless of medium. (Computer source code is also considered to be a literary work.)
- ▹ Musical works: Original musical compositions.
- ▹ Sound recordings of musical works. (different licensing)

- ▷ Dramatic works: literary works that direct a performance through written instructions.
- ▷ Choreographic works must be fixed, either through notation or video recording.
- ▷ Pictorial, Graphic and Sculptural (PGS) works: Any two-dimensional or three-dimensional art work
- ▷ Audiovisual works: work that combines audio and visual components. (e.g. films, television programs)
- ▷ Architectural works (copyright only extends to aesthetics)

▷ The categories are interpreted quite liberally (e.g. for computer code).

▷ There are various requirements to make a work copyrightable: it has to

- ▷ exhibit a certain originality (Schöpfungshöhe)
- ▷ require a certain amount of labor and diligence("sweat of the brow" doctrine)

In short almost all products of intellectual work are copyrightable, but this does not mean copyright applies to all those works. Indeed there is a large body of works that are "out of copyright", and can be used by everyone. Indeed it is one of the intentions of intellectual property laws to increase the body of intellectual resources a society a draw upon to create wealth. Therefore copyright is limited by regulations that limit the duration of copyright and exempts some classes of works from copyright (e.g. because they have already been payed for by society).

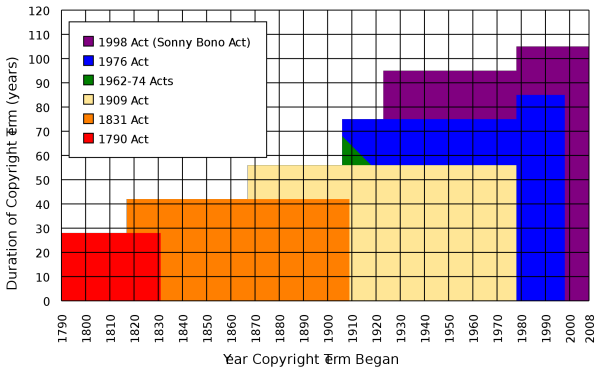
Limitations of Copyrightabilitiy: The Public Domain

▷ **Definition 10.1.7** A work is said to be in the public domain, if no copyright applies, otherwise it is called copyrighted

▷ **Example 10.1.8** Works made by US government employees (in their work time) are in the public domain directly(Rationale: taxpayer already payed for them)

▷ Copyright expires: usually 70 years after the death of the creator

▷ **Example 10.1.9 (US Copyright Terms)** Some people claim that US copyright terms are extended, whenever Disney's Mickey Mouse would become public domain.



Now that we have established, which works are copyrighted — i.e. to which works are intellectual property, let us see who owns them, and how that ownership is established.

Copyright Holder

▷ **Definition 10.1.10** The copyright holder is the legal entity that holds the copyright to a copyrighted work.

▷ By default, the original creator of a copyrightable work holds the copyright.

▷ In most jurisdictions, no registration or declaration is necessary(but copyright ownership may be difficult to prove)

▷ copyright is considered intellectual property, and can be transferred to others (e.g. sold to a publisher or bequeathed)

▷ **Definition 10.1.11 (Work for Hire)** A work made for hire is a work created by an employee as part of his or her job, or under the explicit guidance or under the terms of a contract.

▷ In jurisdictions from the common law tradition, the copyright holder of a work for hires the employer, in jurisdictions from the civil law tradition, the author, unless the respective contract regulates it otherwise.

We now turn to the rights owning a copyright entails for the copyright holder.

Rights under Copyright Law

▷ **Definition 10.1.12** The copyright is a collection of rights on a copyrighted work;

- ▷ personal rights: the copyright holder may
  - ▷ determine whether and how the work is published (right to publish)
  - ▷ determine whether and how her authorship is acknowledged. (right of attribution)
  - ▷ to object to any distortion, mutilation or other modification of the work, which would be prejudicial to his honor or reputation (droit de respect)
- ▷ exploitation rights: the owner of a copyright has the exclusive right to do, or authorize to do any of the following:
  - ▷ to reproduce the copyrighted work in copies (or phonorecords);
  - ▷ to prepare derivative works based upon the copyrighted work;
  - ▷ to distribute copies of the work to the public by sale, rental, lease, or lending;
  - ▷ to perform the copyrighted work publicly;
  - ▷ to display the copyrighted work publicly; and
  - ▷ to perform the copyrighted work publicly by means of a digital-audio transmission.

▷ **Definition 10.1.13** The use of a copyrighted material, by anyone other than the owner of the copyright, amounts to copyright infringement only

when the use is such that it conflicts with any one or more of the exclusive rights conferred to the owner of the copyright.

Again, the rights of the copyright holder are mediated by usage rights of society; recall that intellectual property laws are originally designed to increase the intellectual resources available to society.

Limitations of Copyright (Citation/Fair Use)

▷ There are limitations to the exclusivity of rights of the copyrightholder (some things cannot be forbidden)

▷ Citation Rights: Civil law jurisdictions allow citations of (extracts of) copyrighted works for scientific or artistic discussions.(note that the right of attribution still applies)

▷ In the civil law tradition, there are similar rights:

▷ **Definition 10.1.14 (Fair Use/Fair Dealing Doctrines)** Case law in common law jurisdictions has established a fair use doctrine, which allows e.g.

- ▷ making safety copies of software and audiovisual data
- ▷ lending of books in public libraries
- ▷ citing for scientific and educational purposes
- ▷ excerpts in search engine

Fair use is established in court on a case-by-case taking into account the purpose (commercial/educational), the nature of the work the amount of the excerpt, the effect on the marketability of the work.

10.1.2 Licensing

Given that intellectual property law grants a set of exclusive rights to the owner, we will now look at ways and mechanisms how usage rights can be bestowed on others. This process is called licensing, and it has enormous effects on the way software is produced, marketed, and consumed. Again, we will focus on copyright issues and how innovative license agreements have created the open source movement and economy.

Licensing: the Transfer of Rights

▷ Remember: the copyright holder has exclusive rights to a copyrighted work.

▷ In particular: all others have only fair-use rights(but we can transfer rights)

▷ **Definition 10.1.15** A license is an authorization (by the licensor) to use the licensed material (by the licensee).

▷ Note: a license is a regular contract (about intellectual property) that is

handled just like any other contract. (it can stipulate anything the licensor and licensees agree on) in particular a license may

- ▷ involve term, territory, or renewal provisions
- ▷ require paying a fee and/or proving a capability.
- ▷ require to keep the licensor informed on a type of activity, and to give them the opportunity to set conditions and limitations.

▷ Mass Licensing of Computer Software: Software vendors usually license software under extensive end-user license agreement (EULA) entered into upon the installation of that software on a computer. The license authorizes the user to install the software on a limited number of computers.

Copyright law was originally designed to give authors of literary works — e.g. novelists and playwrights — revenue streams and regulate how publishers and theatre companies can distribute and display them so that society can enjoy more of their work.

With the inclusion of software as "literary works" under copyright law the basic parameters of the system changed considerably:

- modern software development is much more a collaborative and diversified effort than literary writing,
- re-use of software components is a decisive factor in software,
- software can be distributed in compiled form to be executable which limits inspection and re-use, and
- distribution costs for digital media are negligible compared to printing.

As a consequence, much software development has been industrialized by large enterprises, who become copyrights the software was created as work for hire This has led to software quasi-monopolies, which are prone to stifling innovation and thus counteract the intentions of intellectual property laws.

The Free/Open Source Software movement attempts to use the intellectual property laws themselves to counteract their negative side effects on innovation and collaboration and the (perceived) freedom of the programmer.

Free/Open Source Licenses

▷ Recall: Software is treated as literary works wrt. copyright law.

▷ But: Software is different from literary works wrt. distribution channels (and that is what copyright law regulates)

▷ In particular: When literary works are distributed, you get all there is, software is usually distributed in binary format, you cannot understand/cite/-modify/fix it.

▷ So: Compilation can be seen as a technical means to enforce copyright. (seen as an impediment to freedom of fair use)

▷ Recall: IP laws (in particular patent law) was introduced explicitly for two things

- ▷ incentivize innovation (by granting exclusive exploitation rights)

▷ spread innovation (by publishing ideas and processes)

Compilation breaks the second tenet (and may thus stifle innovation)

▷ Idea: We should create a public domain of source code

▷ **Definition 10.1.16** Free/Libre/Open-Source Software (FLOSS) is software that is and licensed via licenses that ensure that its source is available.

▷ Almost all of the Internet infrastructure is (now) FLOSS; so are the Linux and Android operating systems and applications like OpenOffice and The GIMP.

The relatively complex name Free/Libre/Open Source comes from the fact that the English[1] word "free" has two meanings: free as in "freedom" and free as in "free beer". The initial name "free software" confused issues and thus led to problems in public perception of the movement. Indeed Richard Stallman's initial motivation was to ensure the freedom of the programmer to create software, and only used cost-free software to expand the software public domain. To disambiguate some people started using the French "libre" which only had the "freedom" reading of "free". The term "open source" was eventually adopted in 1998 to have a politically less loaded label.

The main tool in brining about a public domain of open-source software was the use of licenses that are cleverly crafted to guarantee usage rights to the public and inspire programmers to license their works as open-source systems. The most influential license here is the Gnu public license which we cover as a paradigmatic example.

GPL/Copyleft: Creating a FLOSS Public Domain?

▷ Problem: How do we get people to contribute source code to the FLOSS public domain?

▷ Idea: Use special licenses to:

- ▷ allow others to use/fix/modify our source code (derivative works)
- ▷ require them to release their modifications to the FLOSS public domain if they do.

▷ **Definition 10.1.17** A copyleft license is a license which requires that allows derivative works, but requires that they be licensed with the same license.

▷ **Definition 10.1.18** The General Public License (GPL) is a copyleft license for FLOSS software originally written by Richard Stallman in 1989. It requires that the source code of GPL-licensed software be made available.

▷ The GPL was the first copyleft license to see extensive use, and continues to dominate the licensing of FLOSS software.

▷ FLOSS based development can reduce development and testing costs (but community involvement must be managed)

[1] the movement originated in the USA

▷ Various software companies have developed successful business models based on FLOSS licensing models. (e.g. Red Hat, Mozilla, IBM, ...)

Note: that the GPL does not make any restrictions on possible uses of the software. In particular, it does not restrict commercial use of the copyrighted software. Indeed it tries to allow commercial use without restricting the freedom of programmers. If the unencumbered distribution of source code makes some business models (which are considered as "extortion" by the open-source proponents) intractable, this needs to be compensated by new, innovative business models. Indeed, such business models have been developed, and have led to an "open-source economy" which now constitutes a non-trivial part of the software industry.

With the great success of open-source sofware, the central ideas have been adapted to other classes of copyrightable works; again to create and enlarge a public domain of resources that allow re-use, derived works, and distribution.

Open Content via Open Content Licenses

▷ Recall: FLOSS licenses have created a vibrant public domain for software.

▷ How about: other copyrightable works: music, video, literature, technical documents

Definition 10.1.19 The Creative Commons licenses are

▷
- ▷ a common legal vocabulary for sharing content
- ▷ to create a kind of "public domain" using licensing
- ▷ presented in three layers (human/lawyer/machine)-readable

▷ Creative Commons license provisions(`http://www.creativecommons.org`)

- ▷ author retains copyright on each module/course
- ▷ author licenses material to the world with requirements

+/- attribuition (must reference the author)
+/- commercial use (can be restricted)
+/- derivative works (can allow modification)
+/- share alike (copyleft) (modifications must be donated back)

10.2 Information Privacy

Information/Data Privacy

▷ **Definition 10.2.1** The principle of information privacy comprises the idea

that humans have the right to control who can access their personal data when.

▷ Information privacy concerns exist wherever personally identifiable information is collected and stored – in digital form or otherwise. In particular in the following contexts

- ▷ Healthcare records
- ▷ Criminal justice investigations and proceedings
- ▷ Financial institutions and transactions
- ▷ Biological traits, such as ethnicity or genetic material
- ▷ Residence and geographic records

▷ Information privacy is becoming a growing concern with the advent of the Internet and search engines that make access to information easy and efficient.

▷ The "reasonable expectation of privacy" is regulated by special laws.

▷ These laws differ considerably by jurisdiction; Germany has particularly stringent regulations (and you are subject to these.)

Acquisition and storage of personal data is only legal for the purposes of the respective transaction, must be minimized, and distribution of personal data is generally forbidden with few exceptions. Users have to be informed about collection of personal data.

Organizational Measures or Information Privacy (under German Law)

▷ Physical Access Control: Unauthorized persons may not be granted physical access to data processing equipment that process personal data.(⇝ locks, access control systems)

▷ System Access Control: Unauthorized users may not use systems that process personal data (⇝ passwords, firewalls, . . .)

▷ Information Access Control: Users may only access those data they are authorized to access. (⇝ access control lists, safe boxes for storage media, encryption)

▷ Data Transfer Control: Personal data may not be copied during transmission between systems (⇝ encryption)

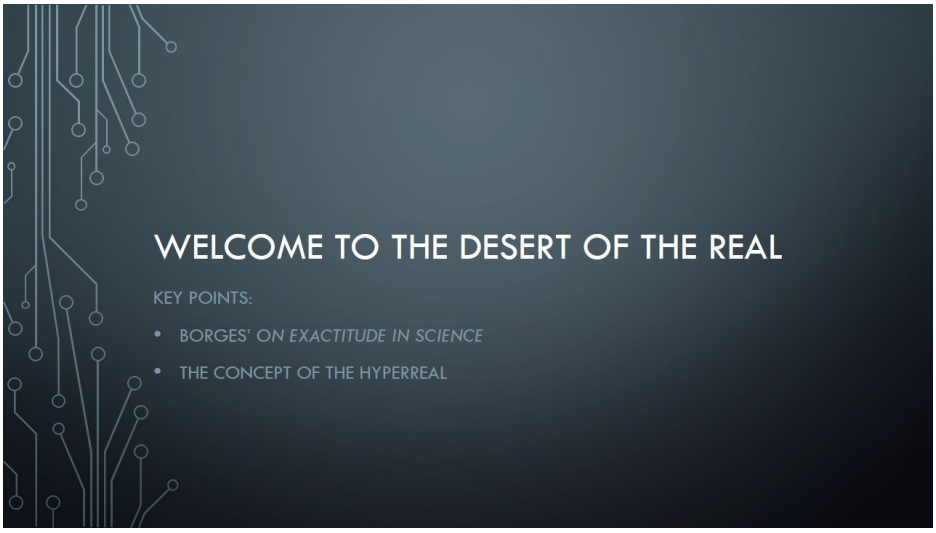
▷ Input Control: It must be possible to review retroactively who entered, changed, or deleted personal data. (⇝ authentification, journaling)

▷ Availability Control: Personal data have to be protected against loss and accidental destruction (⇝ physical/building safety, backups)

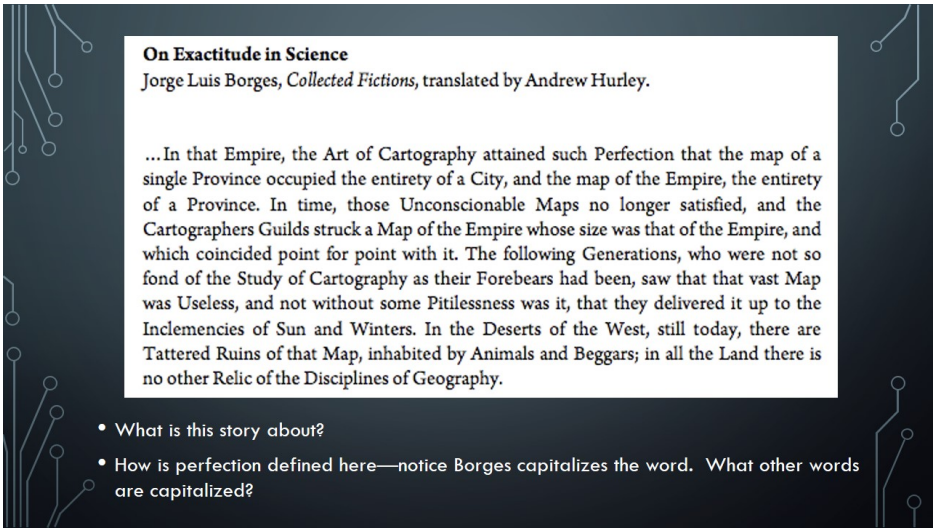
▷ Obligation of Separation: Personal data that was acquired for separate purposes has to be processed separately.

Chapter 11

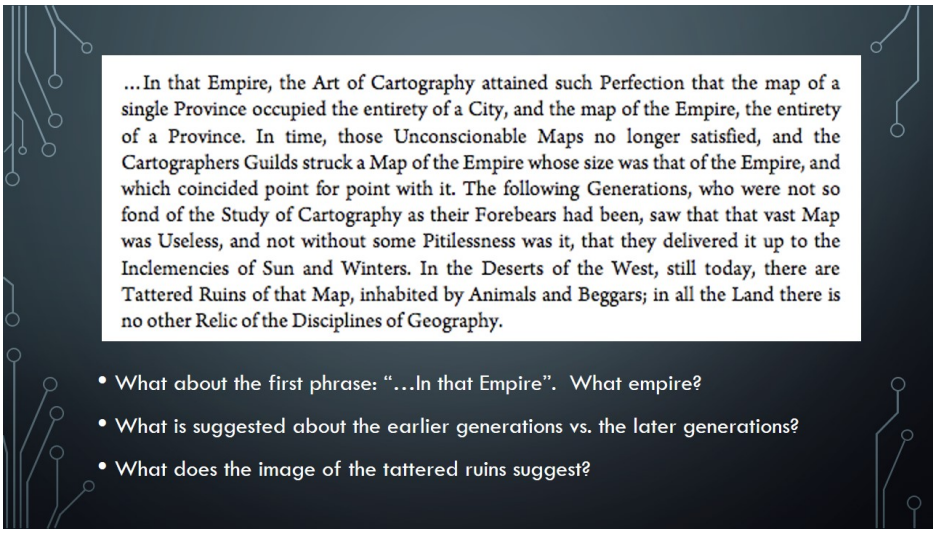
Welcome to the Desert of the Real



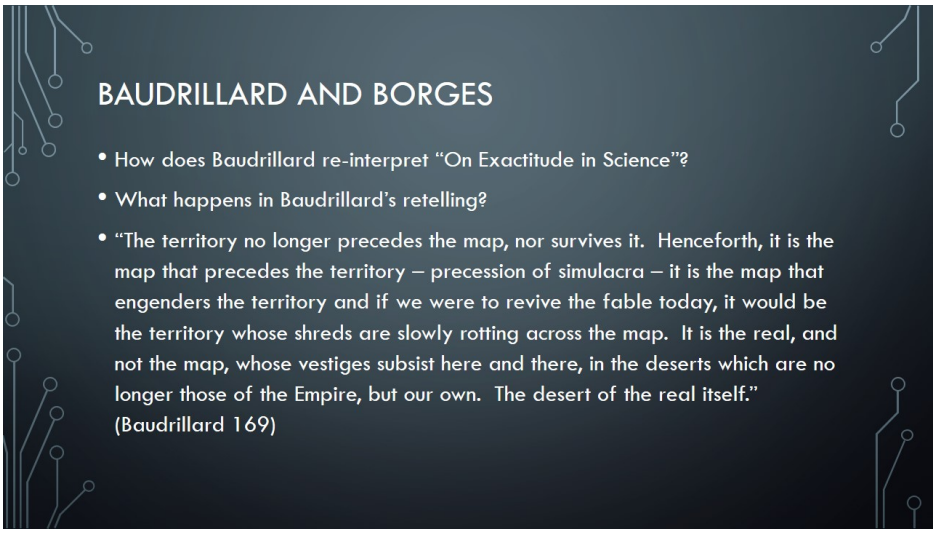
Slide 147



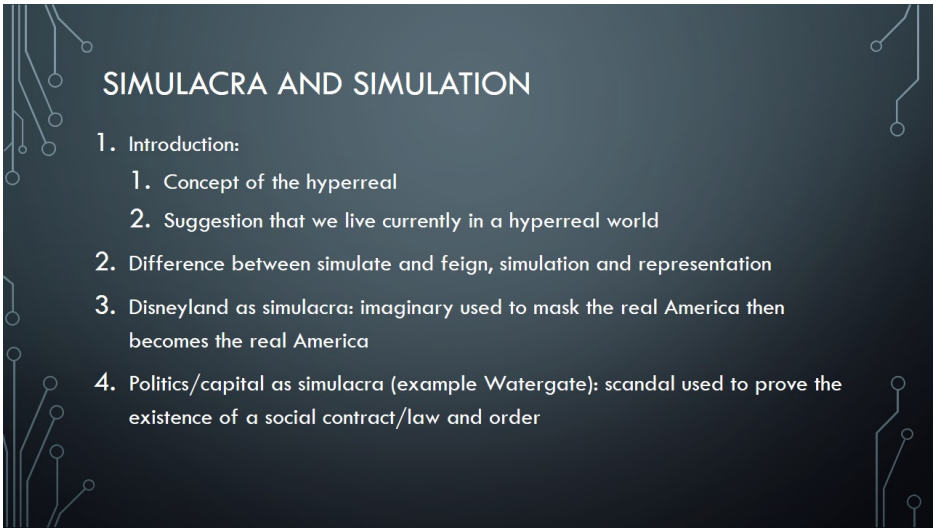
Slide 148



Slide 149



Slide 150



Slide 151

SIMULACRA AND SIMULATION

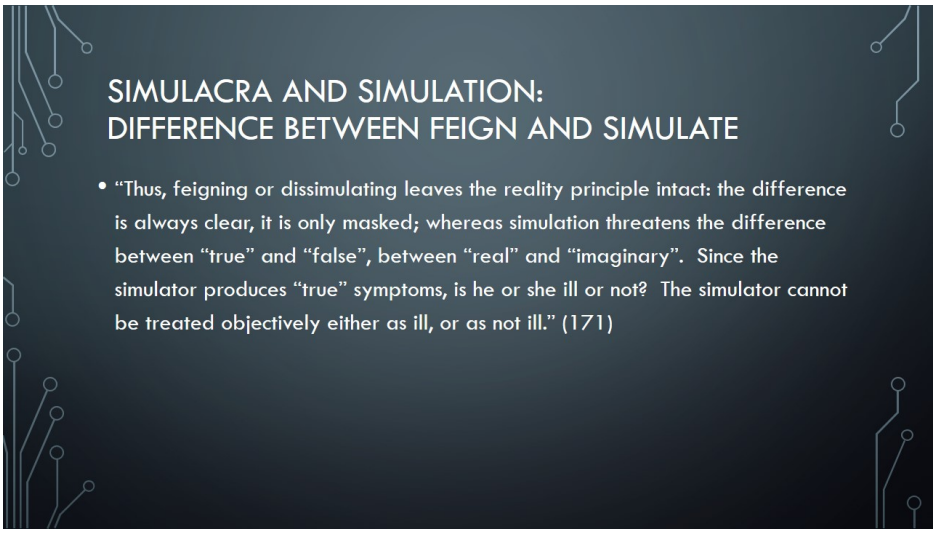
5. The hyperreal as infinite spiral
    1. The Moebius strip: real/imaginary/simulation spiral together
    2. Power (and other forces) uses opposition, crisis, negativity to regenerate itself in the spiral of the hyperreal
6. The loss of the real:
    1. Where/how it dissolves among/into simulation
    2. Consequences of the loss of the real: power/capital/social seek to fall back on the real but end up back in simulation because they cannot be distinguished

Slide 152

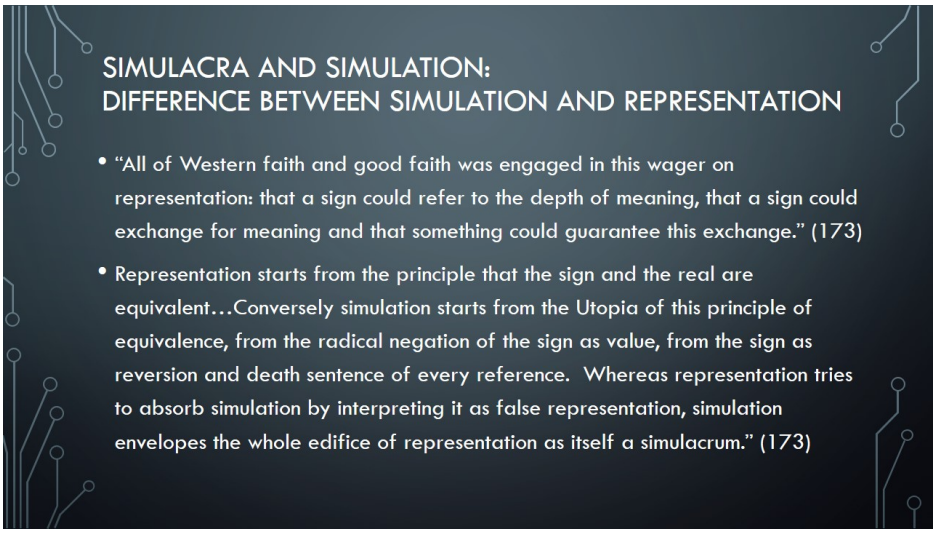
THE HYPERREAL

- "It is a hyperreal: the product of an irradiating synthesis of combinatory models in a hyperspace without atmosphere." (170)
- "It is a question of substituting signs of the real for the real itself…" (170)
- "A hyperreal henceforth sheltered from the imaginary, and from any distinction between the real and the imaginary, leaving room only for the orbital recurrence of models and the simulated generation of difference." (170)

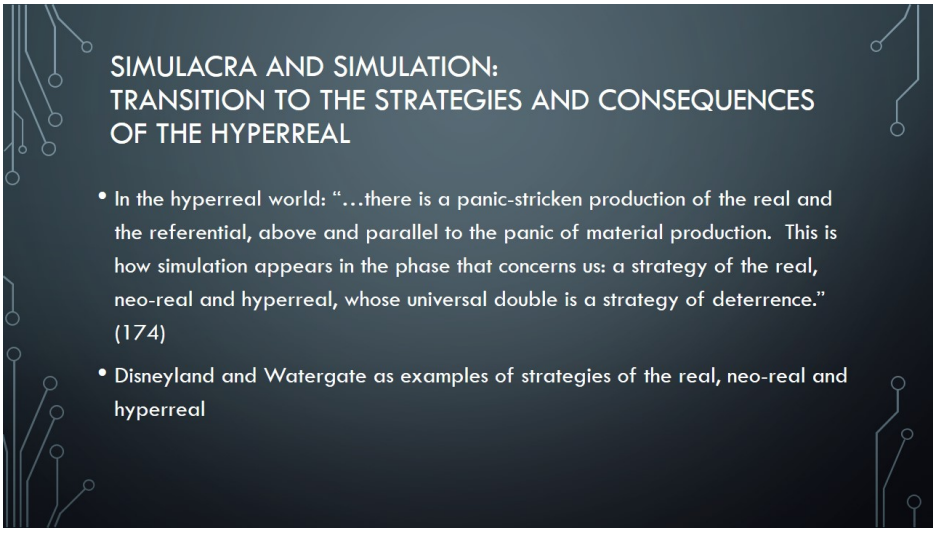
Slide 153



Slide 154



Slide 155



Slide 156

SIMULACRA AND SIMULATION: THE HYPERREAL AS INFINITE SPIRAL

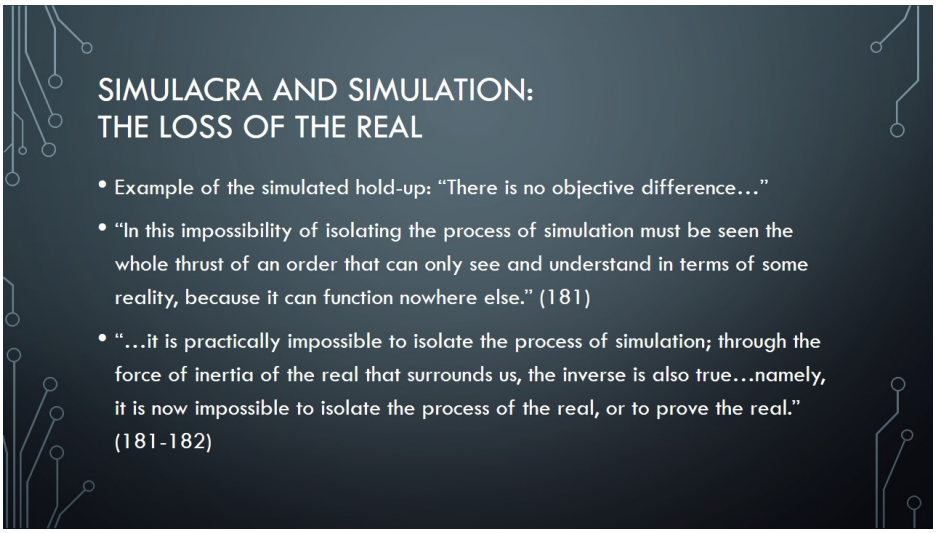
- The Moebius strip: http://www.youtube.com/watch?v=BVslAa2XNKc
- "As for the Moebius strip, if it is split in two, it results in an additional spiral without there being any possibility of resolving its surfaces (hence the reversible continuity of hypotheses)." (179)
- "All the referentials intermingle their discourses in a circular Moebian compulsion." (179)

Slide 157

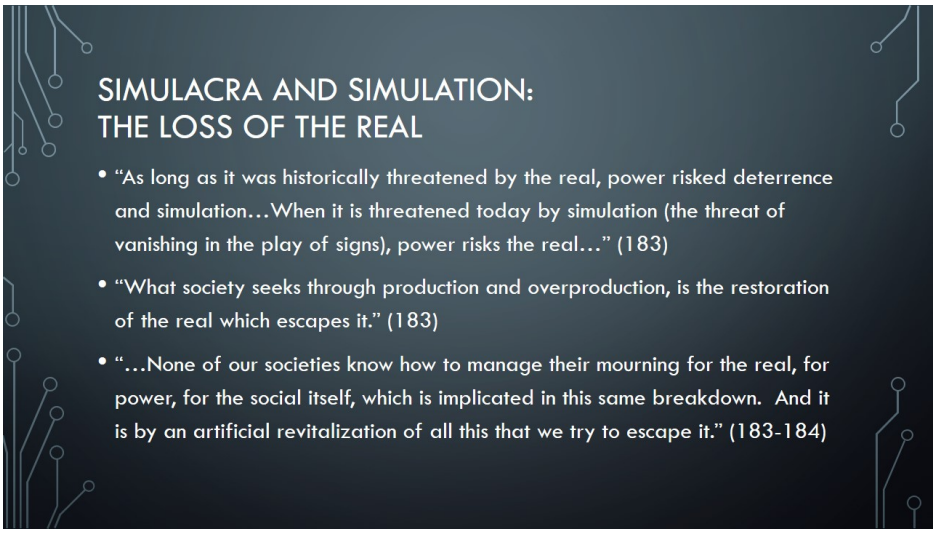
SIMULACRA AND SIMULATION: THE HYPERREAL AS INFINITE SPIRAL

- "It is always a question of proving the real by the imaginary: proving truth by scandal; proving the law by transgression; proving work by the strike; proving the system by crisis and capital by revolution." (179-180)
- "To seek new blood in its own death, to renew the cycle by the mirror of crisis, negativity and anti-power: this is the only alibi of every power, of every institution attempting to break the vicious cycle of its irresponsibility and its fundamental nonexistence, of its deja-vu and its deja-mort." (180)
- How does this relate to our understanding of binary oppositions through deconstruction?

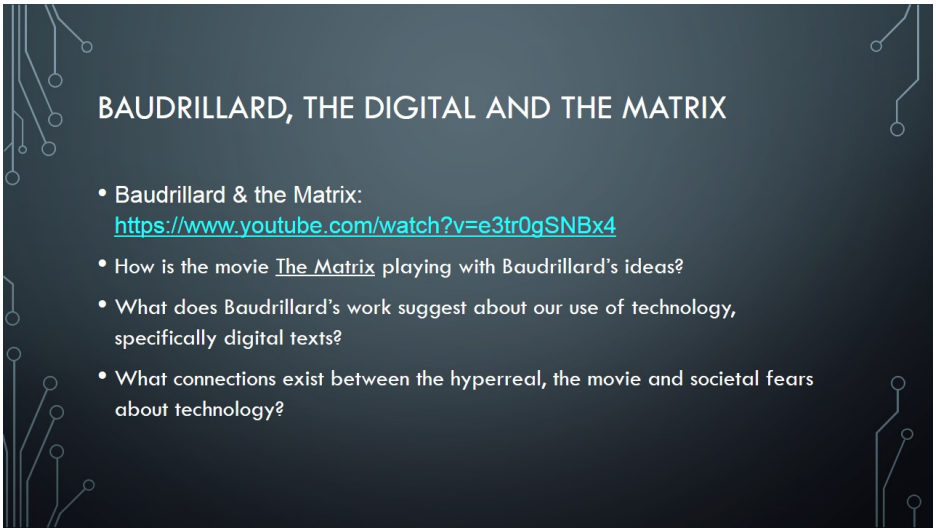
Slide 158



Slide 159



Slide 160



Slide 161

Chapter 12

Computing with Documents

Regular Expressions

▷ **Definition 12.0.2** A regular expression (also called regexp) is a formal expression that specifies a set of strings.

▷ **Definition 12.0.3 (Meta-Characters for Regexps)**

char	denotes
.	any single character
^	beginning of a string
$	end of a string
[...]	any single character in the brackets
[^ ...]	any single character not in the brackets
(...)	marks a group
$\backslash n$	the n^{th} group
|	disjunction
$*$	matches the preceding element zero or more times
$+$	matches the preceding element one or more times
?	matches the preceding element zero or one times
$\{n,m\}$	matches the preceding element between n and m times

▷ **Example 12.0.4 (Regular Expressions and their Values)**

regexp	values
`car`	`car`
`.at`	`cat`, `hat`, `mat`, ...
`[hc]at`	`cat`, `hat`, ...
`[^c]at`	`hat`, `mat`, ... (but not `cat`)
`^[hc]at`	`hat`, `cat`, but only at the beginning of the line
`[0-9]`	Digits
`[1-9][0-9]*`	natural numbers
`(.*)\1`	`mama`, `papa`, `wakawaka`
`cat|dog`	`cat`, `dog`

▷ A regular expression can be interpreted by a regular expression processor (a program that identifies parts that match the provided specification) or a compiled by a parser generator.

Playing with Regular Expressions

▷ If you want to play with regexps, go e.g. to `http://regexpal.com`

The sed Stream Editor

▷ **Definition 12.0.5** The sed utility is a stream editor, it takes a stream (think file) and some regexp replacement commands as an input and gives a stream as a output.

▷ **Example 12.0.6** A sed command is of the form

- ▷ s/⟪regexp⟫/⟪replacement⟫/ (replace once), or
- ▷ s/⟪regexp⟫/⟪replacement⟫/g (replace globally).

▷ To invoke sed in a shell (e.g. on `linux`, `MacOSX`, or `cygwin` on `Windows`)

```
sed -e 's/oldstuff/newstuff/g' inputFileName > outputFileName
```

or (if sedfile.**sed** contains many sed commands)

sed −f sedfile.**sed** inputFileName > outputFileName

▷ **Example 12.0.7 (Update the Jacobs Web Site)**

sed −e 's/International Univ/Jacobs Univ/g;s/IUB/Jacobs/g' index.html > index.html

▷ **Example 12.0.8 (Stalin eliminates Trotzki)** Let cleanse.**sed** be the sed file

```
s/Leon Trotzki//g;s/Trotzki//g
s/Lev Davidovich Bronstein//g;s/Davidovich//g;s/Bronstein//g
```

then Stalin can just use the following shell script to cleanse Kreml documents

find / −name −E ''.∗\.html|.∗\.txt'' −**exec** '**sed** −f cleanse.**sed** {} > {} \;

The lex/flex Lexer Generator

▷ **Definition 12.0.9** The lex is a generator of lexical analyzers (lexers), i.e. a program that reads a lexer specification and outputs C code for a lexer.

A lexer specification is a list of pairs $\langle R, P\rangle$, where R is a regexp and P is C code to be executed when R is matched.

lex is part of UNIX (proprietary), it is extended by the open-source flex.

▷ **Example 12.0.10 (Spotting Integers)**

```
-?[1-9][0-9]* {printf("Saw an integer: %s\n", yytext)}
.|\n { /* Ignore all other characters. */ }
```

If this input is given to flex, it will be converted into a $CLanguage$ file, lex.yy.c. This can be compiled into an executable which matches and outputs strings of integers. For example, given the input abc123z.&*2ghj-6! the program will print:

```
Saw an integer: 123
Saw an integer: 2
Saw an integer: -6
```


lex Example: Tokenizing Artithmetic Expressions

▷ **Example 12.0.11** We want to build a simple calculator, so we need a tokenizer for arithmetic expressions. Here is the flex code for one (see [Vol11] for details):

```
delim          [ \t]
whitesp        {delim}+
digit          [0-9]
number         [-]?{digit}*[.]?{digit}+
%%
{number}  { sscanf(yytext, "%lf", &yylval); return NUMBER;}
"+"       { return PLUS; }
"-"       { return MINUS; }
"/"       { return SLASH; }
"*"       { return ASTERISK; }
"("       { return LPAREN; }
")"       { return RPAREN; }
"\n"      { return NEWLINE; }
{whitesp} { /* No action and no return */}
```

- ▷ The declarations before the %% are abbreviations for number(note that they are recursive)
- ▷ instead of printing notifications we just return token types (values are in yytext)

The yacc/bison Parser Generator

▷ **Definition 12.0.12** yacc (Yet Another Compiler Compiler) is a parser generator, i.e. a program that reads a parser specification and outputs C code for a parser. Historically, yacc was used to generate the C parser in UNIX, today, it is superseded by open-source extensions, e.g. bison.

A yacc parser specification consists of three parts divided by %%.

1. token definitions that specify which tokens to expect from flex
2. grammar and the actions: $$ is the constructed result.
3. more C code, including the usual main function.

yacc/bison Example: Building a Calculator

▷ **Example 12.0.13** We want to build a simple calculator, so we need a tokenizer for arithmetic expressions. Here is the yacc code for one (see [Vol11] for details):

```
 %token NEWLINE NUMBER PLUS MINUS SLASH ASTERISK LPAREN RPAREN
%%
input:                /* empty string */
     | input line;
 line: NEWLINE
     | expr NEWLINE            { printf("\t%.10g\n",$1); };
expr: expr PLUS term          { $$ = $1 + $3; }
     | expr MINUS term         { $$ = $1 - $3; }
     | term;
term: term ASTERISK factor    { $$ = $1 * $3; }
     | term SLASH factor       { $$ = $1 / $3; }
     | factor;
factor:  LPAREN expr RPAREN   { $$ = $2; }
       | NUMBER;
%%
int main(void) {yyparse();exit(0)}
```

Using this to generate a parser with bison gives a program tcalc which is a simple calculator

```
-1.1 + 2 * ( 4 / 3 )
        1566666667
2+2
        4
```


The perl Programming Language

▷ **Definition 12.0.14** perl is a high-level, general-purpose, interpreted, dynamic programming language that makes extensive use of regular expressions.

▷ `perl` can directly use sed commands(with more regexps and execute subroutines)

▷ instead of specifying the language, let us go through an example!

`perl` Example: Correcting and Anonymizing Documents

▷ **Example 12.0.15** We write an a program that makes simple corrections on documents and also crossres out all names.

▷ *The worst president of the US,arguably was George W. Bush. right?*

▷ *However,are you famILIar with Paul Erdős or Henri Poincaré?*(Unicode)

Here is the program:

▷ we first initialize and load modules

```
#!/usr/bin/perl -w
use warnings;
use utf8;
use Encode;
```

▷ then we decode the argument and put it into a variable

```
my $expr = shift;
$expr = decode('utf8',$expr);
```

▷ We put put a space after a comma,

```
$expr =~ s/,(\S)/, $1/g;
```

▷ next we make abbreviations for regular expressions to save space

```
$c=qr/\p{UpperCase_Letter}/;
$l=qr/\p{Lowercase_Letter}/;
```

▷ capitalize the first letter of a new sentence,

```
$expr =~ s/([?.!])\s($l)/$1." ".uc($2)/eg;
```

▷ remove capital letters in the middle of words

```
$expr =~ s/($l)($c+)($l)/$1.lc($2).$3/eg;
```

▷ and we cross-out for official public versions of government documents,

```
$expr =~ s/($c$l+ ($c$l*(\.?) )?$c$l+)/'X' x length($1)/eg;
```

▷ finally, we print the result

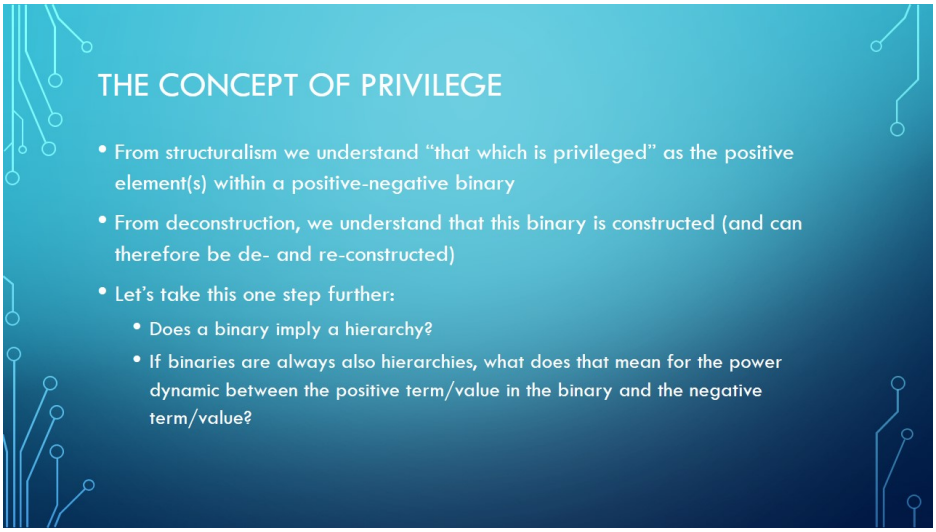
```
print $expr,"\n";
```

The worst president of the US,arguably was George W. Bush. right?
becomes
The worst president of the US, arguably was XXXXXX XX XXXX right?

Chapter 13

Privilege, Language, and the Ditgital

Slide 171

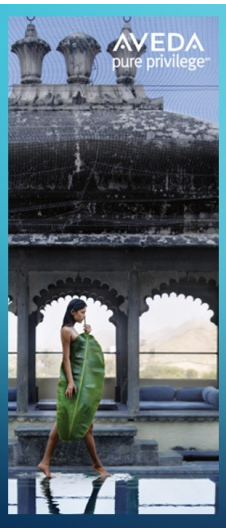


Slide 172

THE CONCEPT OF PRIVILEGE

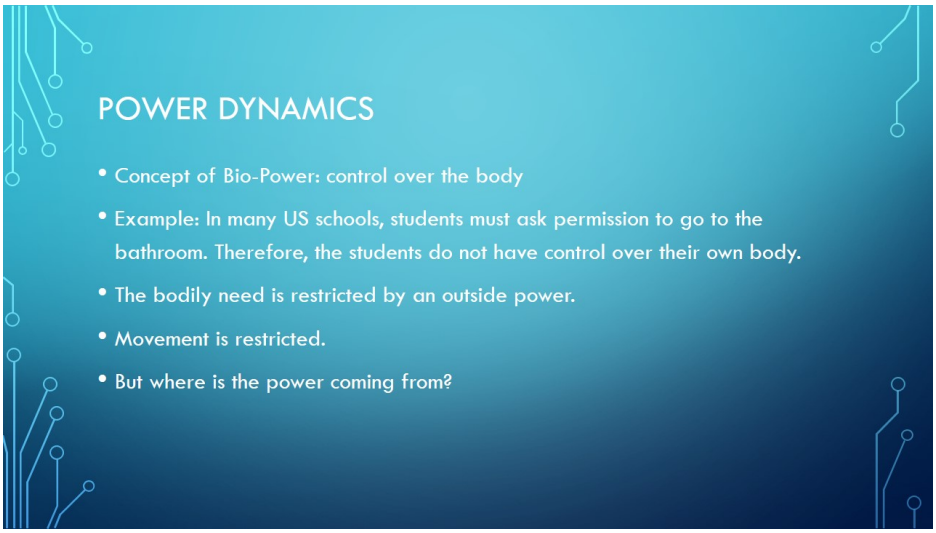
- The white privilege checklist (from a US context): http://www.amptoons.com/blog/files/mcintosh.html
- There are many different forms of privilege: we all have privilege and need to work to recognize this fact
- Privilege always implies and invokes its binary (that which is disparaged)
- Because of the way binaries are aligned, privilege tends to go hand in hand with power (both are the positive terms in their own binaries, privilege/disadvantage and power/weakness

Slide 173

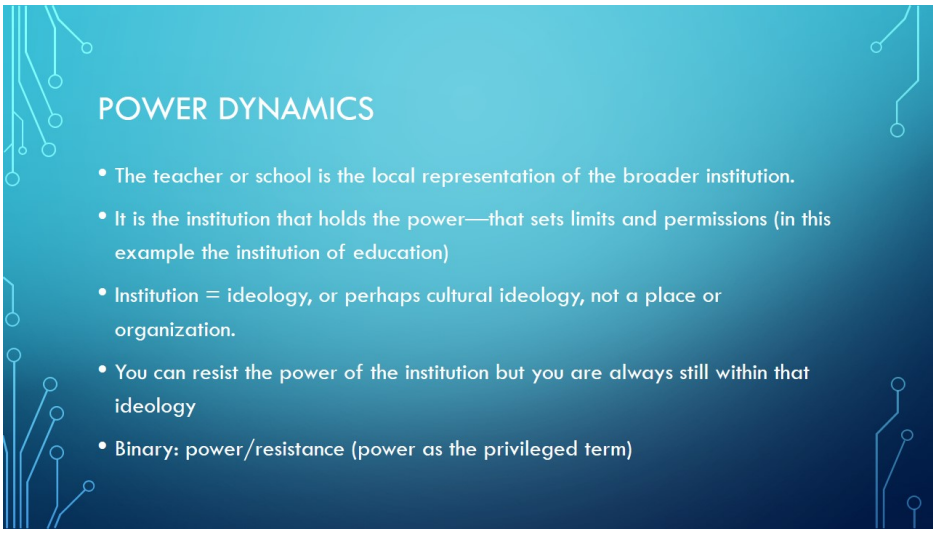


- What does "pure privilege" mean in this advertisement?
- What is being privileged?
- What are the opposing values that are not seen within the image?
- How do we understand the omissions?

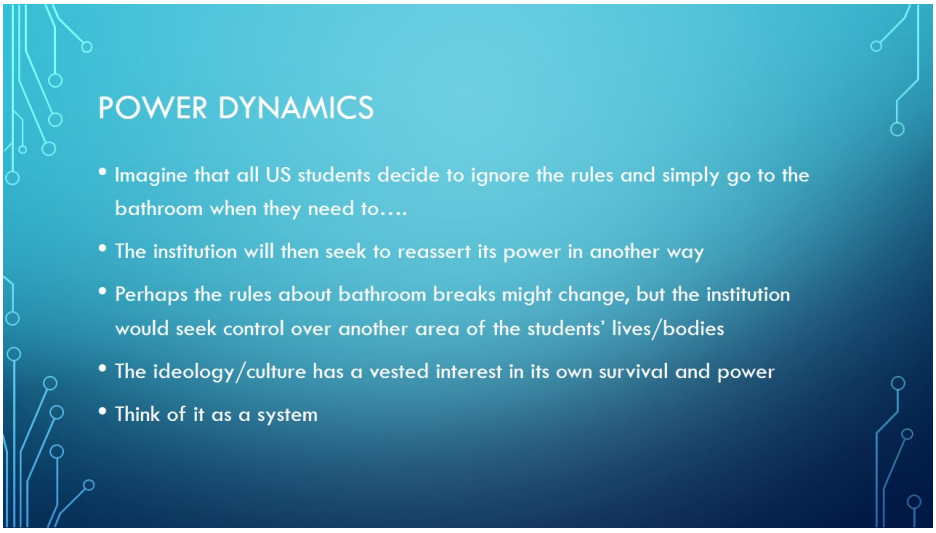
Slide 174



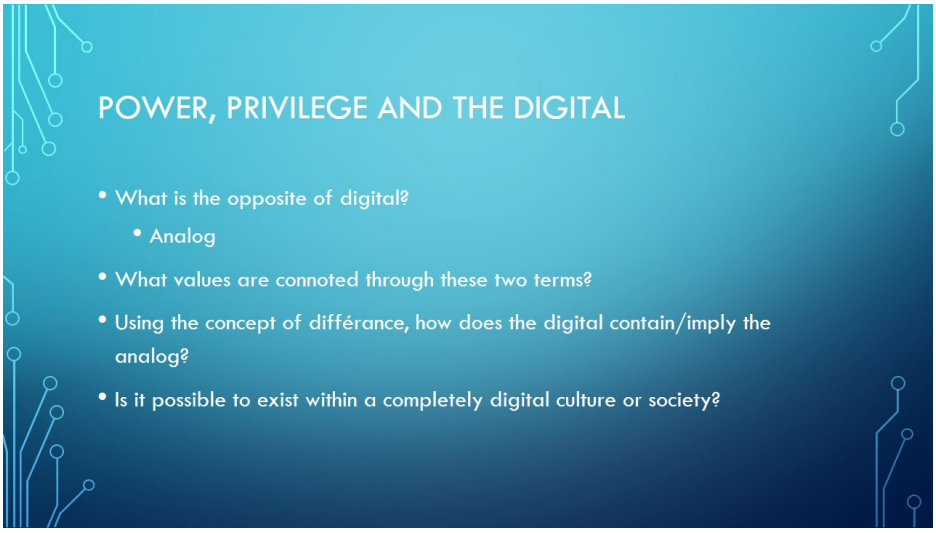
Slide 175



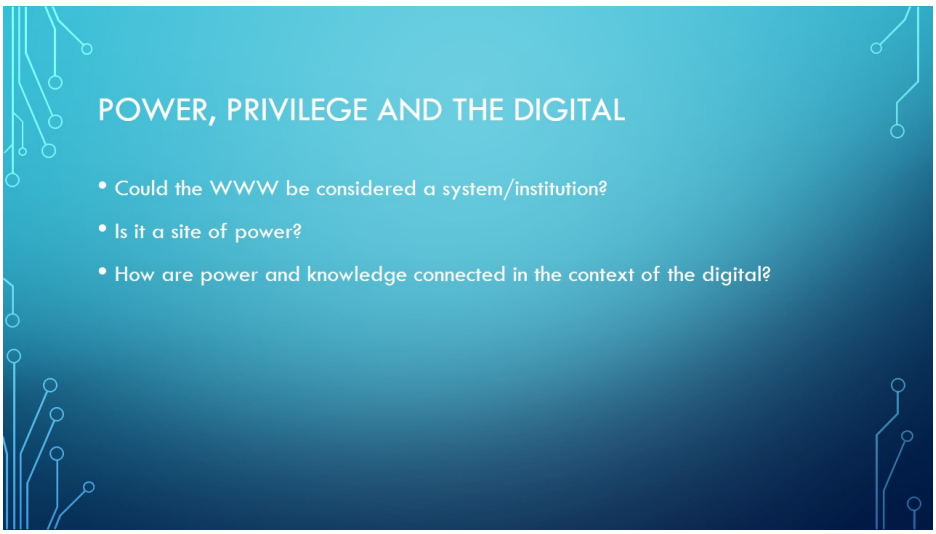
Slide 176



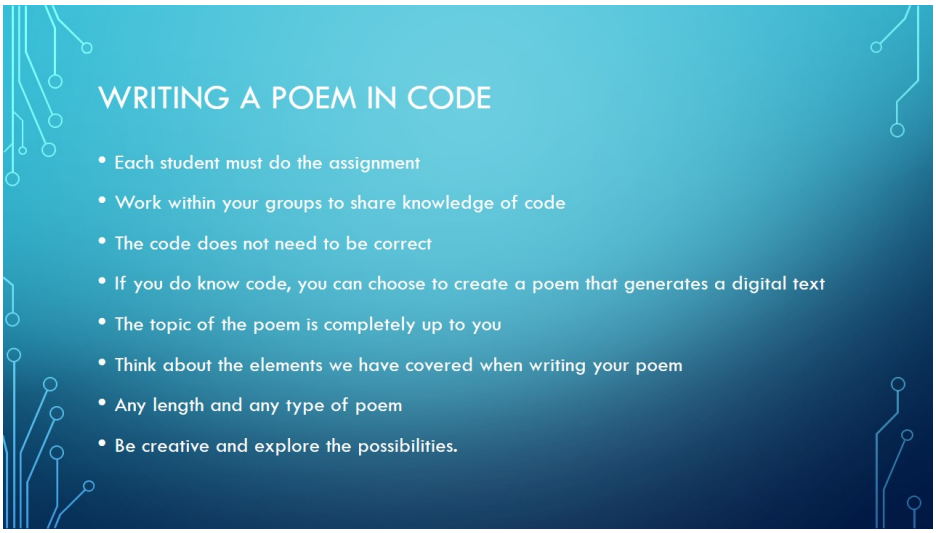
Slide 177



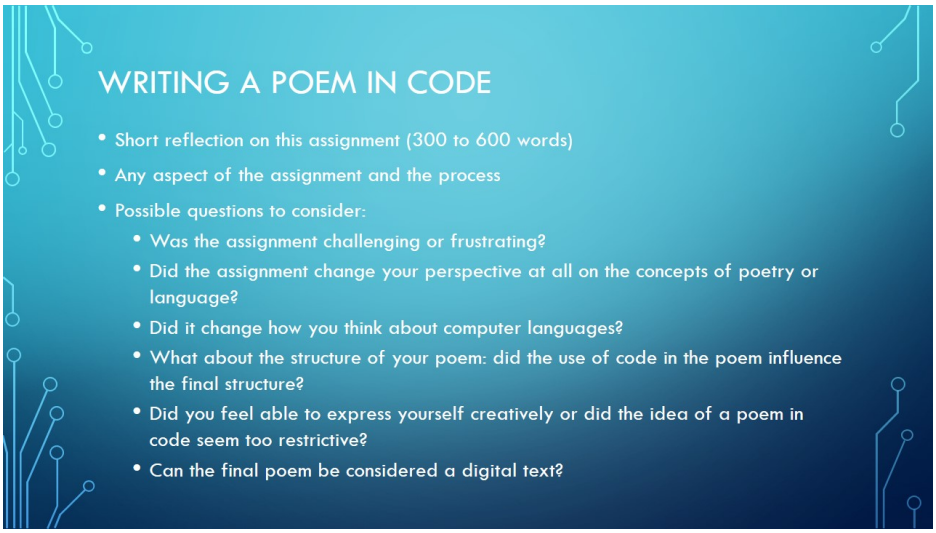
Slide 178



Slide 179



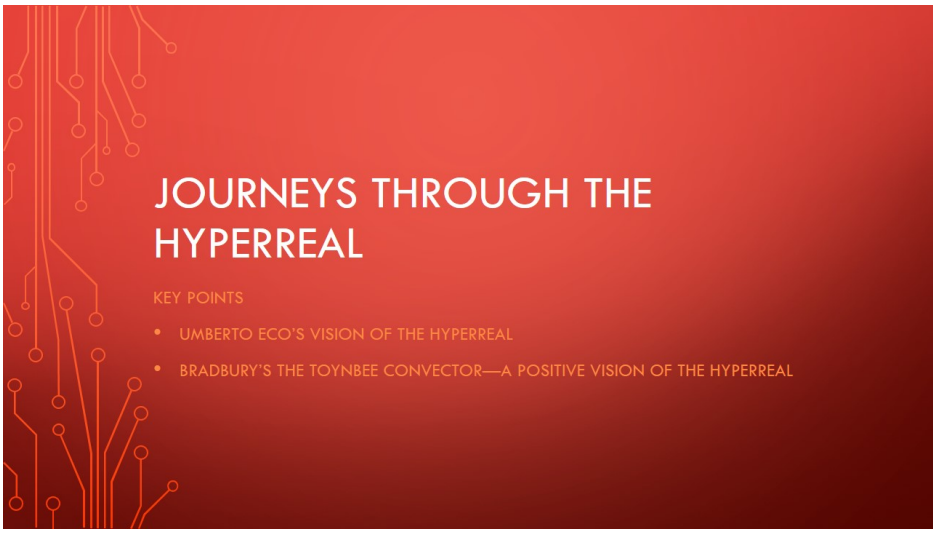
Slide 180



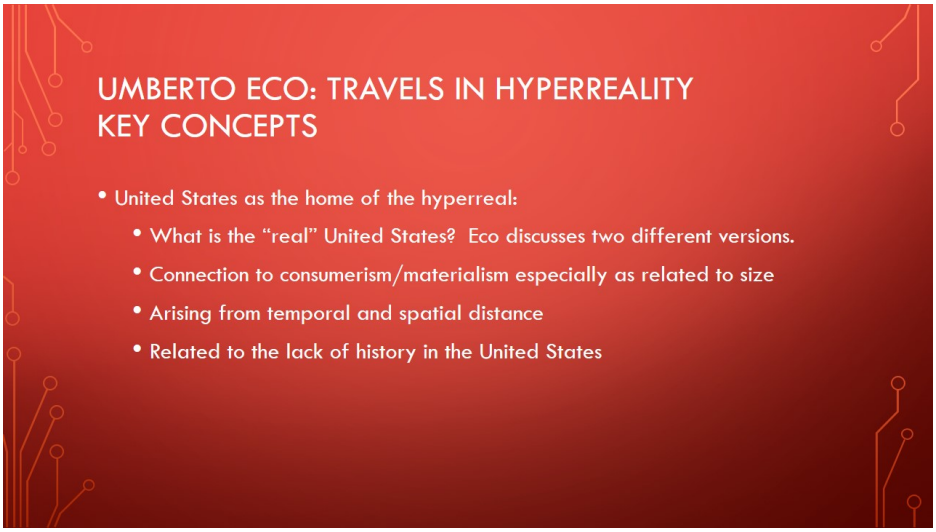
Slide 181

Chapter 14

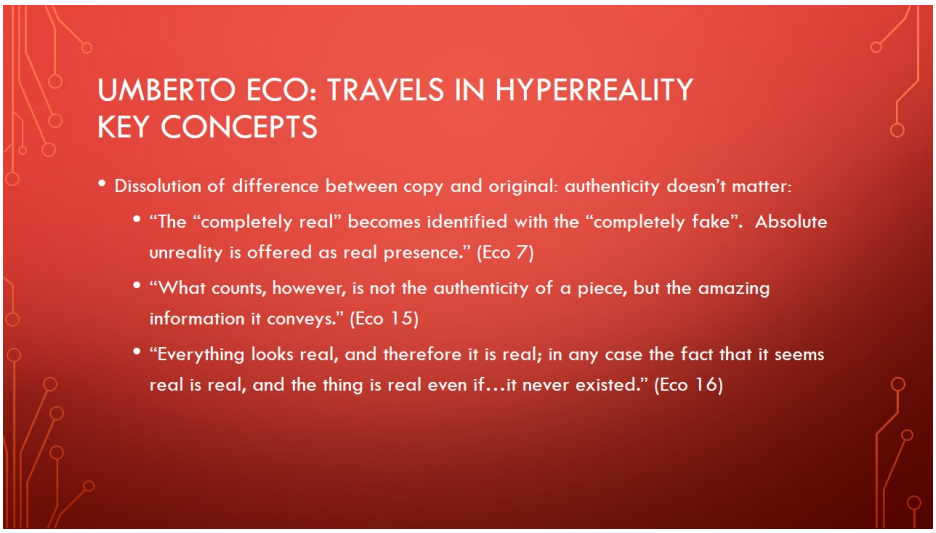
Journeys in the Hyperreall



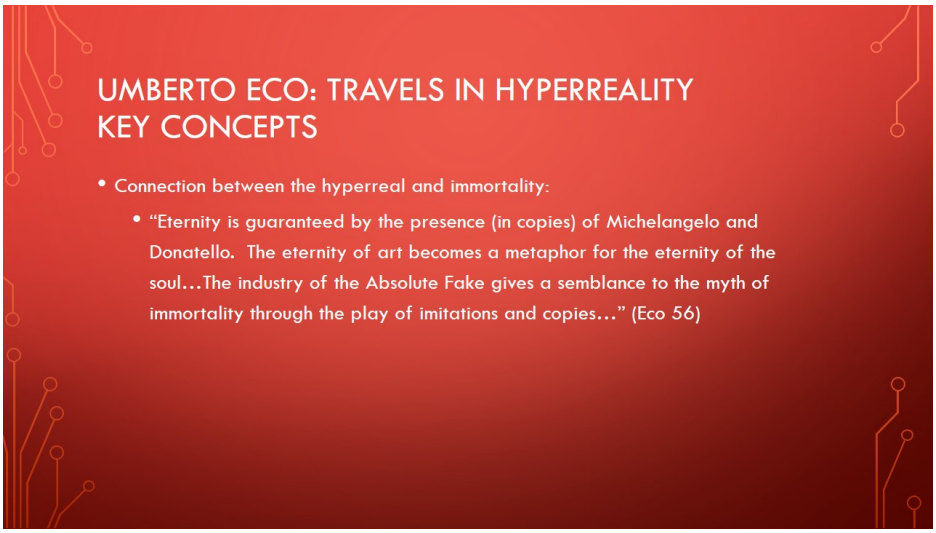
Slide 182



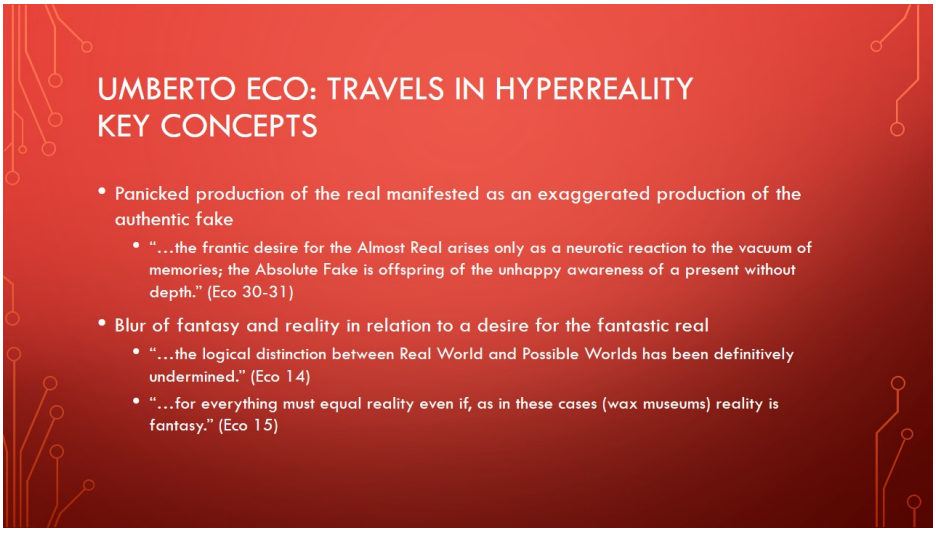
Slide 183



Slide 184



Slide 185



Slide 186

UMBERTO ECO: TRAVELS IN HYPERREALITY KEY CONCEPTS

- Fake or imitation is better/more satisfying than the original or the "true"
  - "The Palace's philosophy is not, 'We are giving you the reproduction so that you will want the original,' but rather, 'We are giving you the reproduction so you will no longer feel any need for the original.' But for the reproduction to be desired, the original has to be idolized…" (Eco 19)
  - "…we not only enjoy a perfect imitation, we also enjoy the conviction that imitation has reached its apex and afterwards reality will always be inferior to it." (Eco 46)

Slide 187

UMBERTO ECO: TRAVELS IN HYPERREALITY DISCUSSION QUESTIONS

- This essay was first written in 1967: what about now? Could the US still be considered the home of the hyperreal? What about Europe and other parts of the world?
- "…the themes of the Last Beach, the apocalyptic philosophy that more or less explicitly rules these reconstructions: Europe is declining into barbarism and something has to be saved." (Eco 36)– Do you think this philosophy is still at work? What is being saved? Something real? How does this relate to Baudrillard?
- How do the concepts of Empire and imperialism play into both Eco and Baudrillard?

Slide 188

DISNEYLAND/DISNEYWORLD AS SIMULACRA

Umberto Eco:

- Emphasizes the authenticity of the consumer experience
- "…Disneyland makes it clear that within its magic enclosure it is fantasy that is absolutely reproduced…But once the 'total fake' is admitted, in order to be enjoyed, it must seem totally real." (Eco 43)
- "…Disneyland not only produces illusion, but in confessing it—stimulates the desire for it." (Eco 44)
- Disneyland is the paean of the hyperreal

Baudrillard:

- "Disneyland is presented as imaginary in order to make us believe that the rest is real, when in fact all of Los Angeles and the America surrounding it are no longer real, but of the order of the hyperreal and simulation. It is no longer a question of false representation of reality (ideology), but of concealing the fact that the real is no longer real, and thus of saving the reality principle." (Baudrillard 175)
- Example: Adults enter the childlike world of Disneyland, which obsucres their own childishness and that of the outside world

Slide 189

EPCOT AND THE HYPERREAL

- Neither Baudrillard or Eco address directly Epcot at Walt Disney World and Epcot's "around the world"
- Full scale buildings mimic actual landmarks/architecture
- People working there with nametags that say where they are from (i.e. Montpelier, Nanjing, etc.)
- How do we define the experience for someone who has never left the US? How about for someone who has travelled?
- How do we understand Epcot through the ideas of Baudrillard and Eco?

Slide 190

BRADBURY'S: THE TOYNBEE CONVECTOR

- Eco: "…and this tells us a lot about the ravenous consumption of the present and about the constant "past-izing" process carried out by American civilization in its alternate process of futuristic planning and nostalgic remorse." (Eco 9-10)
- Bradbury: "Life has always been lying to ourselves. As boys, young men, old men. As girls, maidens, women, to gently lie and prove the lie true." (Bradbury 8)

Slide 191

BRADBURY'S: THE TOYNBEE CONVECTOR

- Why does Shumway discard the evidence of the time traveler's illusion?
- What does this story suggest about the power of the hyperreal? How is it different from Eco's and Baudrillard's perspectives?

Slide 192

THE HYPERREAL AND THE DIGITAL

- "Disneyland tells us that technology can give us more reality than nature can." (Eco 44)
- Given everything we have discussed in regards to the hyperreal, how can we understand this quote from Eco in relation to modern technology, and specifically digital texts?
- Assignment: Examples of the hyperreal

Slide 193

Chapter 15

Programming Documents

Idea: Even though documents should be thought of as sequences of characters with markup (and images, formulae, tables, etc.), we can also think of them as *programs that produce such characters with markup*. In some situations, this is profitable, e.g. when the documents have parts that can be computed from the rest, e.g. a table of contents, the section numberings, or indices. In such situations, the author does not need to type in the computable document fragments, but can just represent them by a command. A conversion program interprets such a "document program" (usually text interspersed with commands), executes all the commands, and outputs a document (without commands), which can then be read. The main advantage of the "documents as programs" paradigm is that the computed document fragments can never get out of sync with the rest of the document, which eases the maintenance burden over the document life-cycle.

There are various implementations of this idea, in this chapter we present the TEX/LATEX system, in which the `pdflatex` program is used to transform documents with macros into `PDF`. Systems like PHP do similar things for the Web.

The TEX Typesetting System

▷ **Definition 15.0.16** Typesetting is the process of creating the visual appearance of a document by assembling glyphs (visual representations of characters; also called types) on pages.

▷ Since Gutenberg's time (to ca. 1975), typesetting was done by assembling movable types (special metal positives of single letters) into lines and later into pages, which were inked and the printed; or using negatives to form cast-metal positives for printing.

▷ **Definition 15.0.17** TEX is a typesetting program designed by Donald Knuth in 1978. It combines movable types (character boxes) with macro programming.

▷ **Definition 15.0.18** The `pdftex` program reads a file of text marked up with TEX macros and outputs `PDF`.

▷ **Example 15.0.19 (Hello World in TEX)** `pdftex` typesets the following TEX file

```
Hello, World \bye
```

The command sequence \bye stops pdftex and is not shown in the output.

Note that the "document program"

```
Hello, World \bye
```

the pdftex interprets all characters as "self-inserting characters", i.e the character "a" is essentially a command that inserts a character "a" into the PDF (in the right font and size).

We have already seen one document program command used by TEX above, and there are many more. Most of them insert special characters into the document or change the formatting. But TEX goes much further, it allows the author to define commands as well. This makes the TEX format self-extensible, and into a very expressive special purpose programming language for documents.

TEX Macros for Programming Documents

- ▷ TEX uses command sequences (words starting with "\"; also called macros) for special effects.
- ▷ **Example 15.0.20** \bye stops the formatter, \alpha prints α, \int prints $\int$,...
- ▷ Users can also define TEX macros as abbreviations via \def
- ▷ **Example 15.0.21** \def\tdm{Text and Digital Media} defines the macro \tdm.
  We love the USC ``\tdm''! expands to
  "We love the USC "Text and Digital Media"!
- ▷ TEX macros can have arguments specify with #1, #2...: delimit with { and }
- ▷ **Example 15.0.22** with the macro \def\tnwhat#1{Text and \textbf{#1}} \tnwhat{Beer} expands to "Text and **Beer**"

TEX was invented by a mathematician, so it is not a surprise that it is the most capable tool for typesetting formulae — an art that only a select few professional typesetters (humans who put lead into rows) could do.

Mathematical Formulae in TEX

- ▷ **Definition 15.0.23** TEX has a math mode for formulae delimited with $ (inline math) or \[and \] (display math)
- ▷ **Example 15.0.24** Some TEX commands can be used everywhere: e.g. the Greek letters, \alpha prints α, \beta prints β,...
- ▷ **Example 15.0.25** Many TEX commands only make sense in math mode: e.g. superscripts with ^, e.g. x^3 gives x^3, subscripts with _, e.g. x_{ij} gives x_{ij}, \int prints $\int$, \frac{1}{2} prints $\frac{1}{2}$,...
- ▷ **Example 15.0.26** $\int_0^\infty f(\theta) d\theta$ expands to $\int_0^\infty f(\theta)d\theta$

▷ **Example 15.0.27** Use macros in math mode as well: `\def\frac#1#2{#1\over #2}`
Then `\[1+\frac{2}{2+\frac{3}{3+\ldots}}\]` expands to

$$1+\frac{2}{2+\frac{3}{3+\ldots}}$$

One of the things that TEX is useful for is to automate numbering of sections, subsections, footnotes, etc. For that TEX offers some basic data structures. Here we introduce counters, and show how we can make simple sectioning macros from them.

TEX Counters

▷ TEX uses special macros as counters, `\newcount`, allocates a counter, `\advance` alters it, and `\the` references it.

▷ **Example 15.0.28** We define a sectioning macros

```
\newcount\seccount % allocate a new counter for sections
\newcount\subseccount % allocate a new counter subsections
\seccount0\subseccount0 % initialise both with 0
\def\section#1{ % begin macro definition
\advance\seccount by 1 % step the counter
\subseccount0 % reset the subsection counter
\textbf{\Large\the\seccount. #1} % section number and title
} % end macro definition
\def\subsection#1{\advance\subseccount by 1
\textbf{\large\the\seccount.\the\subseccount. #1}}
```


Anyone who is experienced in programming realizes that TEX is not a modern programming language. But of course, it was conceived in 1978, the age of COBOL, and a lot has happened in programming language design since then. But even if it is relatively inconvenient and ugly code, it gets the job done.

We will now present a couple of internal macros that build up to more document automation that shows the advantages of programming documents: a serial letter macro.

TEX Conditionals

▷ TEX provides some conditionals for your use:
e.g. `\ifx` compares two macros, `\ifnum` compares two number, and `\ifmmode` tells you if you are in math mode.
`\if⟨cond⟩...\else...\fi` uses it.

▷ TEX uses special macros for user-defined conditionals, `\newif\if⟨cond⟩`, allocates a conditional, `⟨cond⟩true` and `⟨cond⟩false` alter it,

Programming a Chain Letter

▷ **Example 15.0.29 (A Parametric Reminder)**

```
\def\reminder#1#2{\hfill Bremen, \today\par\bigskip
\noindent Dear #1,\par\medskip\noindent
please be sure that you will not forget to come to the lecture
today. We are planning big things.\par\medskip\noindent
Sincerely,\par\bigskip\noindent #2\newpage}
```

▷ **Example 15.0.30 (Programming a Serial Letter)**
We can use arbitrary characters to delineate arguments in macro definitions.

```
\def\sletter#1,#2;{\def\first{#1}\def\second{#2}\def\empty{}
\ifx\first\empty\else\reminder{#1}{Thomas \& Michael}
\ifx\second\empty\else\sletter#2,;\fi\fi}
\def\serialletter#1{\sletter #1;}
```

Also nothing prevents us from using recursion.

▷ **Example 15.0.31 (Making a Serial Letter)**

```
\serialletter{Mati, Anca, Isabel, Calin}
```


Our serial letter example shows that with a bit of programming effort the self-extensibility of TEX can be used to automate various document-oriented tasks, or style the documents for a given situation. Naturally, this brought forth a vibrant community that started swapping and re-using TEX programs.

TEX Macro Packages

▷ Idea: Separate out common macro definitions into a separate file and include that via `\input`. (So we can reuse them over multiple documents)

▷ Actually: many people have already done that.

▷ The AMS (American Mathematical Society) supplies AMSTEX: TEX macros that make it more convenient to write Math (e.g. the `\frac` macro)

▷ Till Tantau supplies tikz (TEX ist kein Zeichenprogram): TEX macros that allow you to draw images.

▷ Leslie Lamport supplies LATEX, a set of TEX packages and classes. `pdflatex` is `pdftex` with the LATEX package macros pre-loaded.

▷ The bibTEX package handles bibliographic references.

The most widely used macro package for TEX is LATEX, there are tens of thousands of macro packages that use the basic LATEX infrastructure. LATEX is the standard for high-end document

formatting for scientific/technical documents nowadays. We now show a typical document as model for your own documents.

The Anatomy of a LaTeX Document

▷ **Example 15.0.32 (A LaTeX file:** `main.tex`**)**

```
\documentclass{article} % use the article class (Journal Article)
\title{Anatomy of a {\LaTeX} Document} % specify the title,
\author{Michael Kohlhase\\Jacobs University Bremen} % author,
\date{\today} % and date
\begin{document} % start the document
\maketitle % make the title
\tableofcontents % make the table of contents
\section{Introduction}\label{sec:intro}
This is really easy, just start writing,
\section{Main Part}\label{sec:main}
We refer the reader to~\cite{Lamport:ladps94} for details. But there should be at least
one formula: \[1+\frac{2}{2+\frac{3}{3+\ldots}}\]
\section{Conclusion}\label{concl:intro}
As we already said in Section~\ref{sec:intro} on
p. \pageref{sec:intro} this was not so bad was it?
\bibliographystyle{alpha}
\bibliography{example}
\end{document}
```

▷ Format it with `pdflatex main` (generates `main.aux` for references)

and the bibTeX database used in it

▷ **Example 15.0.33 (a bibTeX file** `example.bib`**)**

```
@BOOK{Lamport:ladps94,
  title = {LaTeX: A Document Preparation System, 2/e},
  publisher = {Addison Wesley},
  year = {1994},
  author = {Leslie Lamport}}
```

▷ Generate bibliography with `bibtex main`(it knows about `example.bib` from `main.aux`)

▷ run `pdflatex` twice (to get all the cross-references right)

The Result (generated parts in red)

Anatomy of a LaTeX Document

Michael Kohlhase
Jacobs University Bremen

January 21, 2014

Contents

1. Introduction

This is really easy, just start writing,

2. Main Part

We refer the reader to [Lam84] for details. But there should be at least one formula:

$$1+\cfrac{2}{2+\cfrac{3}{3+\ldots}}$$

3. Conclusion

As we already said in Section 1 on p. 1 this was not so bad was it?

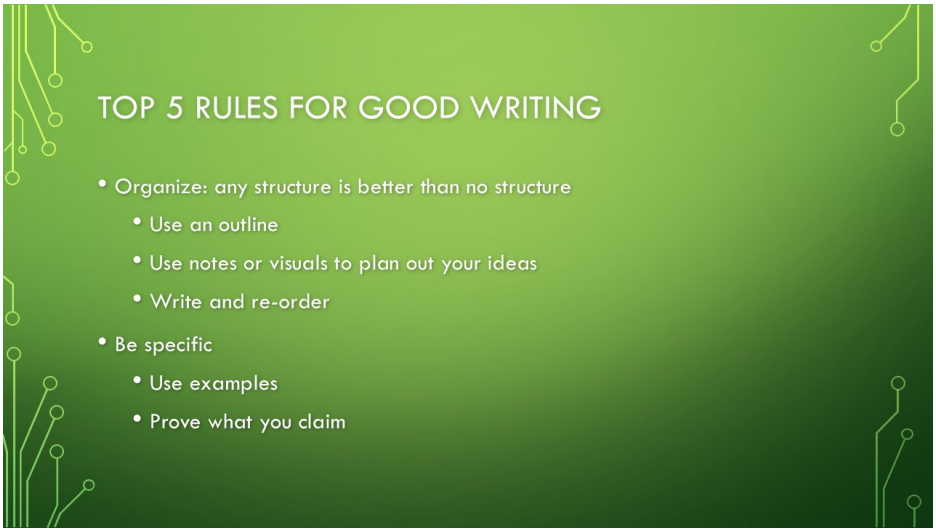
References

[Lam94] Leslie Lamport, *LaTeX: A Document Preparation System, 2/e*, Addison Wesley, 1994.

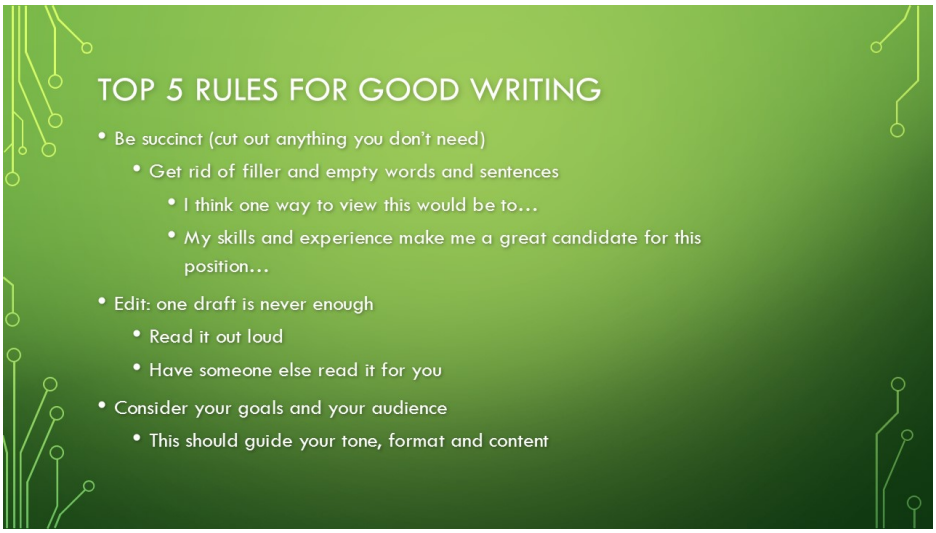
Chapter 16

Practical Writing Tips

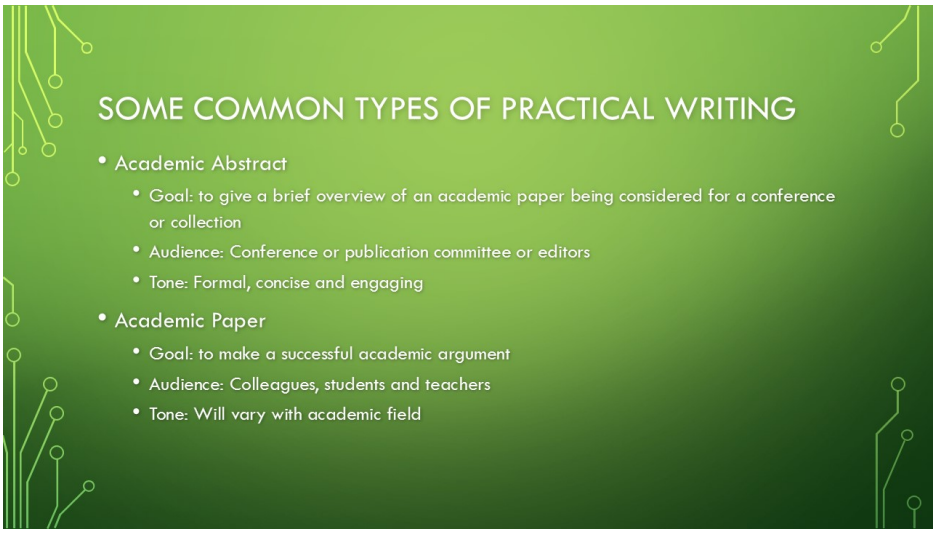
Slide 204



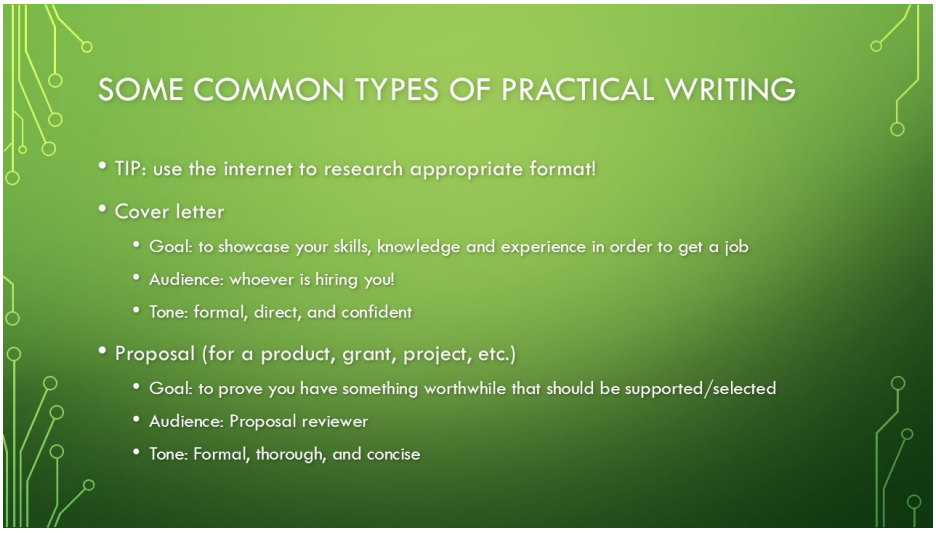
Slide 205



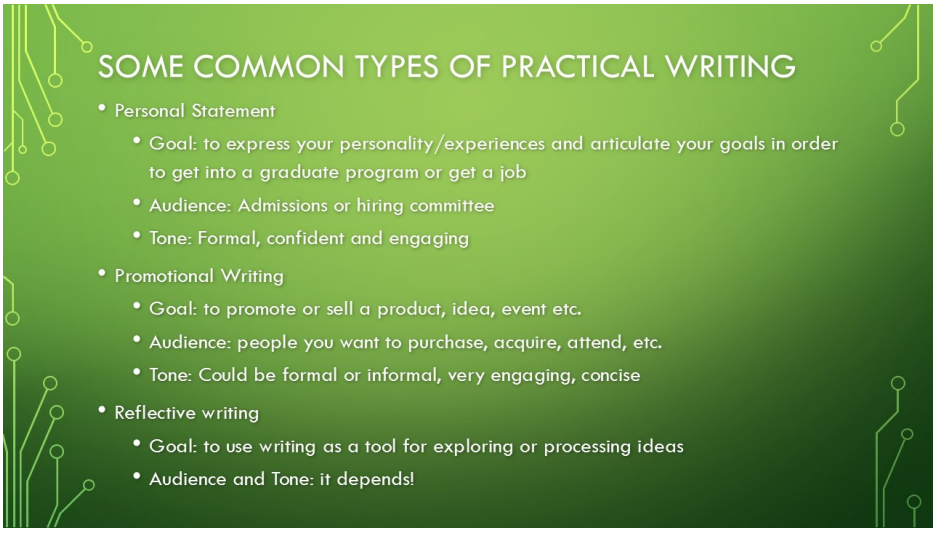
Slide 206



Slide 207



Slide 208



Slide 209

Chapter 17

Electronic Books and their Formats

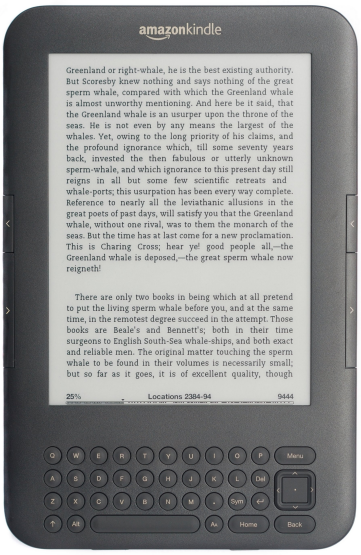
Electronic Books

▷ **Definition 17.0.34** An electronic book (eBook) is a publication in electronic form that can be read on digital devices.

▷ **Example 17.0.35** Arguably the first eBooks were the texts provided by Project Gutenberg in 1971.

▷ **Definition 17.0.36** An electronic book reader (eReader) is a hardware or software devide for reading electronic books.

▷ **Example 17.0.37** Popular hardware-based eReaders are Kindle (Amazon.com), the iPad (Apple), and the Nook (Barnes&Noble), but sofware readers also abound.



EPUB: A Standard for Electronic Publishing [Wik11]

Definition 17.0.38 EPUB is a free and open standard for electronic books provided by the International Digital Publishing Forum (IDPF).
It consists of three specifications:

- **Open Publication Structure** (OPS), essentially XHTML and CSS for the document contents
- **Open Packaging Format** (OPF), which describes the structure of the EPUB file in XML.
- **Open Container Format** (Ocf), which collects all files as a ZIP archive.

▷ EPUB files usually have the extension `.epub`.

▷ EPUB does not specify a format for digital rights management (DRM), which makes it less attractive for the big publishers.

▷ EPUB is supported by almost all eReaders and publishing software

EPUB: Open Packaging Format & Navigation Control

▷ **Definition 17.0.39** The Open Packaging Format (OPF) is a standard for specifying giving additional structure and coherence to an electronic book in EPUB. It specifies the

- ▷ contents (what files) in the `manifest` element
- ▷ metadata (author, date, etc) in the `metadata` element
- ▷ linear reading order in the `spine` element, and
- ▷ (optionally) important structural components in the `guide` element.

of the package in a OPF file with the extension `.opf`.

▷ **Definition 17.0.40** The navigation control of the an EPUB gives a machine-readable table of contents of the book in XML.

An Example OPF file

```
<?xml version="1.0"?>
<package version="2.0" xmlns="http://www.idpf.org/2007/opf" unique-identifier="BookId">

  <metadata xmlns:dc="http://purl.org/dc/elements/1.1/"
                  xmlns:opf="http://www.idpf.org/2007/opf">
    <dc:title>Pride and Prejudice</dc:title>
    <dc:language>en</dc:language>
    <dc:identifier id="BookId" opf:scheme="ISBN">123456789X</dc:identifier>
    <dc:creator opf:file-as="Austen, Jane" opf:role="aut">Jane Austen</dc:creator>
  </metadata>

  <manifest>
    <item id="chapter1" href="chapter1.xhtml" media-type="application/xhtml+xml"/>
    <item id="stylesheet" href="style.css" media-type="text/css"/>
    <item id="ch1-pic" href="ch1-pic.png" media-type="image/png"/>
    <item id="myfont" href="css/myfont.otf" media-type="application/x-font-opentype"/>
```

```
    <item id="ncx" href="book.ncx" media-type="application/x-dtbncx+xml"/>
  </manifest>

  <spine toc="ncx">
    <itemref idref="chapter1" />
  </spine>

  <guide>
    <reference type="loi" title="List Of Illustrations" href="appendix.html#figures" />
  </guide>

</package>
```


An Example NCX file

```
<?xml version="1.0" encoding="UTF-8"?>
<ncx version="2005-1" xml:lang="en" xmlns="http://www.daisy.org/z3986/2005/ncx/">

  <head>
   <meta name="dtb:uid" content="123456789X"/> <!-- same as in .opf -->
   <meta name="dtb:depth" content="1"/> <!-- 1 or higher -->
   <meta name="dtb:totalPageCount" content="0"/> <!-- must be 0 -->
   <meta name="dtb:maxPageNumber" content="0"/> <!-- must be 0 -->
 </head>

  <docTitle>
    <text>Pride and Prejudice</text>
  </docTitle>

  <docAuthor>
    <text>Austen, Jane</text>
  </docAuthor>

  <navMap>
    <navPoint class="chapter" id="chapter1" playOrder="1">
      <navLabel><text>Chapter 1</text></navLabel>
      <content src="chapter1.xhtml"/>
    </navPoint>
  </navMap>

</ncx>
```


EPUB: Open Container Format

▷ **Definition 17.0.41** An EPUB file is a group of files conforming to the OPS/OPF standards that is wrapped in a ZIP file. The Open Container Format (OCF) specifies how these files should be organized in the ZIP archive, and defines two additional files that must be included.

▷ The `mimetype` file must be a text document in ASCII and must contain the string `application/epub+zip`. It must also be uncompressed, unencrypted, and the first file in the ZIP archive.

▷ The purpose of this file is to provide a more reliable way for applications to identify the mimetype of the file than just the `.epub` extension.

▷ Also, there must be a folder named `META-INF` which contains the required file `container.xml`. This XML file points to the file defining the contents of the book. This will be the `.opf` file.

An Example Container

ZIP Container

```
mimetype
META-INF/
container.xml
OPS/
book.opf
book.ncx
chapter1.xhtml
ch1-pic.png
css/
style.css
myfont.otf
```

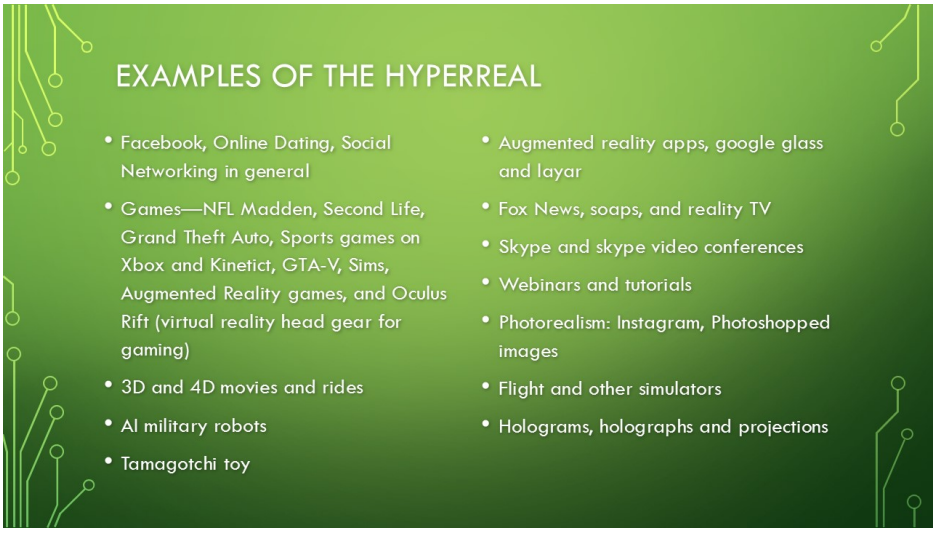
container.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<container version="1.0"
    xmlns="urn:oasis:names:tc:opendocument:xmlns:container">
  <rootfiles>
    <rootfile full-path="OPS/book.opf"
       media-type="application/oebps-package+xml"/>
    <rootfile full-path="OPS/book.ncx"
       media-type="application/x-dtbncx+xml"/>
  </rootfiles>
</container>
```

Chapter 18

I'm So Meta

Slide 217



Slide 218

I'M SO META EVEN THIS ACRONYM

- From the Greek prefix meaning "after/beside/among/with"
- Used in the term "metaphysics" where the prefix means an abstraction about/beyond the root: metaphysics is the study of the abstract concepts about and beyond the physical
- Began to be used in a mathematical concept in the1920's and 1930s when it sometimes (as in the term "metatheorom") took on the modern meaning of: an X about X (self-referential).
- Douglas Hofstader used the term in a 1979 book and most modern usages of the term likely originated from this book.

Slide 219

THE CONCEPT OF META-TEXTS

- Meta-text: a text describing or commenting on another text
- Remember our discussion of texts and digital texts:
  - Images
  - Videos
  - E-books
  - E-mail
  - Webpages and websites (clusters of webpages)
  - Digital documents (i.e. pdf, wiki, google doc)
  - Online databases

Slide 220

THIS SENTENCE CONTAINS THIRTY-SIX LETTERS

Slide 221

MORE EXAMPLES: THAT'S SO META

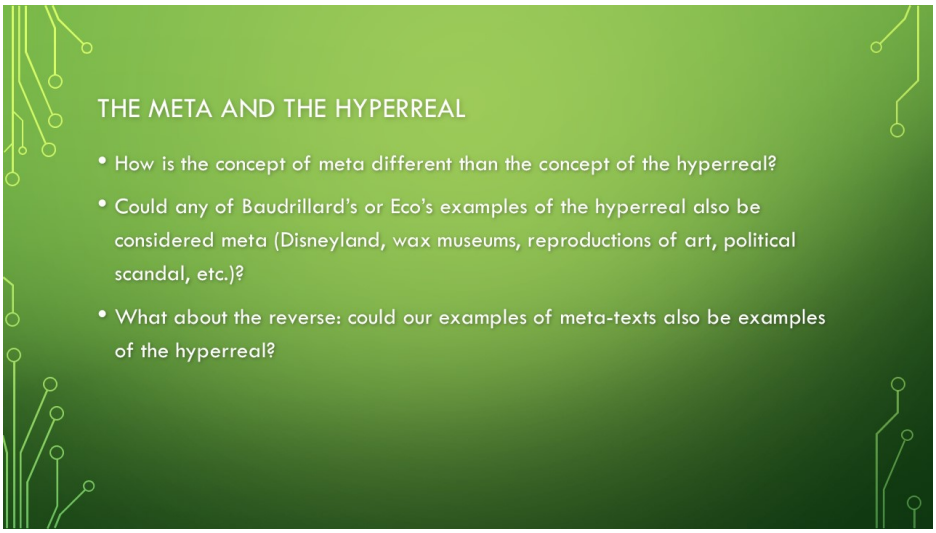
- Their is four errors in this sentance. Can you find them?
- Breaking the fourth wall--Meta-movies:
  - http://flavorwire.com/327251/10-awesome-meta-movies-that-will-melt-your-mind/2/

Slide 222

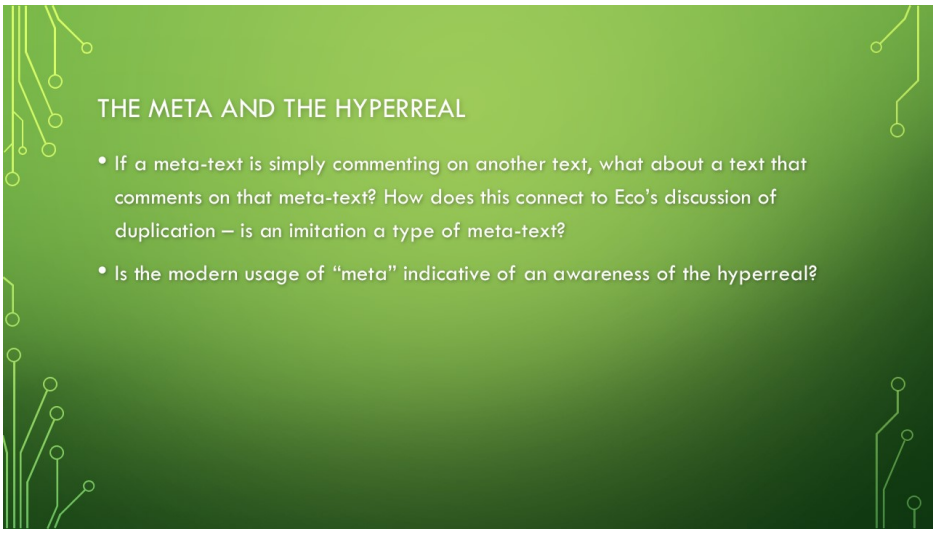
THE META AND THE SELF

- Since meta-texts are self-referential are they then narcissistic? What does it mean to be narcissistic?
- How does the consciousness of a meta-text implicate the self?
  - Consider the viewpoint of the reader of the text, as well as the text itself
- Does the meta simultaneously create distance from self and emphasis on self? If so, why? (think about our understanding of différance)

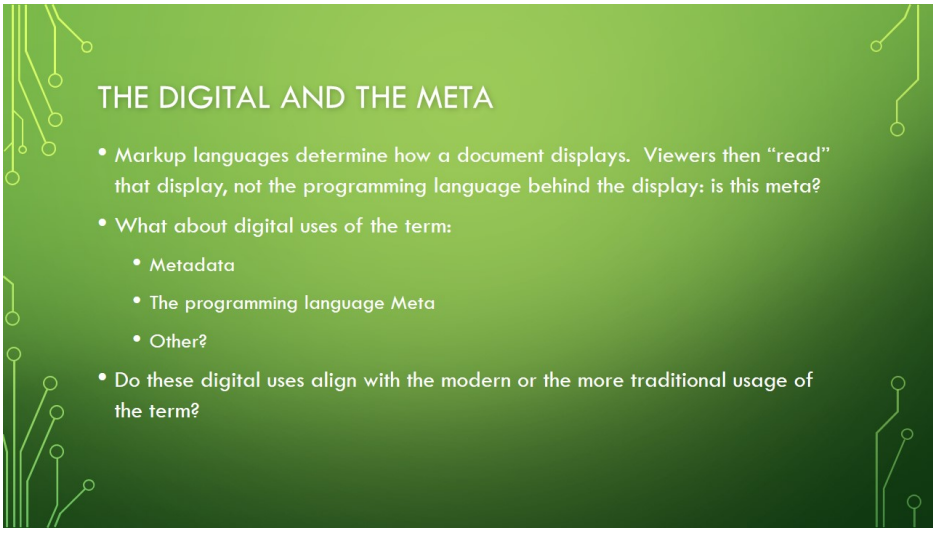
Slide 223



Slide 224



Slide 225



Slide 226

Chapter 19

Writing Technical Documentation and Manuals

19.1 Technical Documentation in DocBook

DocBook

▷ **Definition 19.1.1** DocBook is a content markup language for technical documentation based on SGML or XML. It supplies elements/tags for the logical of book-like documents.

▷ DocBook was originally intended for writing technical documents related to computer hardware and software but it can be used for any other sort of documentation.

▷ DocBook content is presentation-neutral and can be published in a variety of formats, including HTML, XHTML, EPUB, PDF, man pages and HTML Help, without requiring users to make any changes to the source.

▷ DocBook began in 1991 as a joint project of HAL Computer Systems and O'Reilly & Associates. Since 1998 it is maintained by a Technical Committee at OASIS.

DocBook Elements

▷ DocBook provides about 400 content markup tags

▷ Structural Elements: specify broad characteristics of their contents, e.g. book, part, article, chapter, appendix, dedication

▷ Block-level Elements: specify structured blocks of text (usually starting and ending with new "lines"). e.g. paragraphs, lists, definitions, etc. They usually have a fixed content model; some can contain text.

▷ Inline-level Elements: wrap text within a block-level element (usually without breaking "lines"), e.g. for emphasis, hyperlinks, definienda,. They typically cause the document processor to apply some kind of distinct typographical treatment to the enclosed text.

DocBook Example

▷ A "Hello World" document in DocBook

```
<?xml version="1.0" encoding="UTF-8"?>
<book xml:id="simple_book" xmlns="http://docbook.org/ns/docbook" version="5.0">
  <title>Very simple book</title>
  <chapter xml:id="chapter_1">
    <title>Chapter 1</title>
    <para>Hello world!</para>
    <para>
      I hope that your day is proceeding
      <emphasis>splendidly</emphasis>!
    </para>
  </chapter>
  <chapter xml:id="chapter_2">
    <title>Chapter 2</title>
    <para>Hello again, world!</para>
  </chapter>
</book>
```


19.2 Topic-Oriented Documentation with DITA

DITA the "Darwin Information Typing Architecture"

▷ **Definition 19.2.1** DITA is a topic-oriented content markup language for technical documentation based on XML. It supports a topic-oriented documentation style.

▷ **Definition 19.2.2** The basic unit of information in DITA is a topic, i.e. a discrete piece of content that is about a specific subject, has an identifiable purpose, and can stand alone (does not need to be presented in context for the end-user to make sense of the content).

▷ Topics can be reused in any context; DITA makes use of this.

▷ **Definition 19.2.3** DITA combines topics into documents via DITA maps.

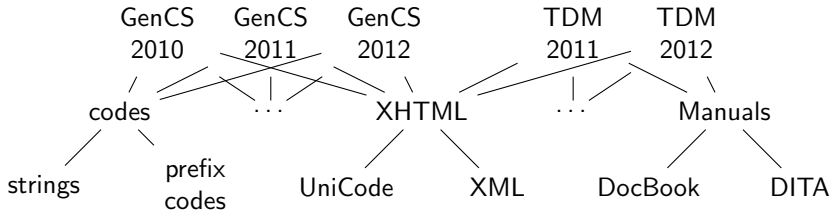
▷ Consequence: A DITA topic (and DITA map) can be referenced in multiple DITA maps.

▷ Extension: Conditional text allows filtering or styling content based on attributes for audience, platform, product, and other properties.(the DITA processor filters text)

Using DITA Maps for Reuse

▷ Idea: Concepts can be reused in more than one DITA map

▷ **Example 19.2.4** For instance a module on HTML/XML in the courses "General Computer Science" and "Text and Digital Media".



Courses given in different years share most of their content (but not all)

A DITA Concept File

▷ **Definition 19.2.5** A DITA concept is a special DITA topic that describes an abstract idea or a named unit of knowledge.

▷ **Example 19.2.6** A concept for "academic conference" (note the conditional text)

```
<concept id="A.dita">
  <title>Academic Conference</title>
  <conbody>
    <p audience="students">
     An <term>academic conference</term> is a gathering of scientists
     who discuss <term>scientific papers</term>.
    </p>
    <p audience="professors">
      An <term>academic conference</term> is a pretense to travel to
      nice locations on university money and drink loads of beer.
    </p>
    <para conref="#topic/p2"/>
  </conbody>
  <related-links>
    <linkpool type="concept">
      <link audience="students" href="http://easychair.org"/>
      <link audience="professors" href="http://acapulco.mx"/>
    </linkpool>
  </related-links>
</concept>
```

We can generate two versions from this content markup format. For instance, with the following DITA value specification:

```
<!-- this file specifies the actions for students -->
<val>
  <prop action="exclude" att="audience" val="professors"/>
  <prop action="include" att="audience" val="students"/>
</val>
```

A DITA Task File

▷ **Definition 19.2.7** A DITA task is a special DITA topic that describes a process.

▷ **Example 19.2.8** DITA task markup for assignment 8 of the TDM course

```
<task id="TDMassignment8">
  <title>Assignment 8: Reviewing Papers</title>
  <taskbody>
    <prereq>You have to be a registered TDM student.</prereq>
   <steps>
      <step>
        <cmd>accept the PC invitation, log into easychair</cmd>
        <info>You should have been given the information in the invitation e-mail</info>
      </step>
      <step>
        <cmd>indicate your conflicts of interest</cmd>
        <info>you have a conflict with anybody you have a relationship that
          would keep you from being objective (yourself, your family members,
          loved/hated ones, group members,... be honorable)
        </info>
        <stepresult>
          <p>The system records a list of conflicted paper and will not show you anything about them.</p>
        </stepresult>
      </step>
    </steps>
  </taskbody>
</task>
```


A DITA Map File

▷ **Definition 19.2.9** A DITA map combines DITA topics and maps into a document by transclusion.

▷ **Example 19.2.10** <map>

```
  <title>Life as an Academic</title>
  <topicmeta>...</topicmeta>
  <topicref href="introduction.dita" collection-type="sequence">
    <topicref href="conference.dita"/>
    <topicref href="TDMassignment8.dita"/>
  </topicref>
  <reltable>
    <relcell>conference.dita</relcell>
    <relcell>TDMassignment8.dita</relcell>
  </reltable>
</map>
```

Chapter 20

Revision Control Systems

We address a very important topic for document management: supporting the document life-cycle as a collaborative process. In this chapter we discuss how we can use a set of tools that have been developed for supporting collaborative development of large program collections can be used for document management.

We will first introduce the problems and current attempts at solutions and the introduce two classes of revision control systems and discuss their paradigmatic systems.

20.1 Introduction/Motivation

Lifecycle Management for Digital Documents

▷ Documents may have a non-trivial life-cycle involving multiple actors.

▷ **Example 20.1.1** For a novel we have the following stages:

1. skeleton/layout (chapters, characters, interactions)
2. first complete draft (given out to test readers)
3. private editing cycle $\rightsquigarrow$ accepted draft(testing with more readers, refining/condensing the story)
4. publisher's editing cycle $\rightsquigarrow$ final draft(professional editor proposes refinements to the draft)
5. copyediting for spelling, adherence of publisher's house style
6. adding artwork/cover $\rightsquigarrow$ first published edition
7. e-dition (eBook) etc. (different artwork, links, interactivity)

▷ **Example 20.1.2** For technical books, multiple editions follow to adapt them to changing domain or correct errors.

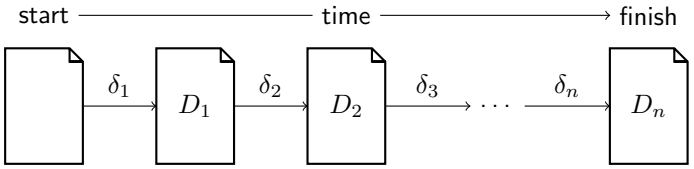
Document Lifecycle Mgmt. & Collaboration Approaches

▷ Practice: Send around MS Word documents by e-mail (dates in file name)

▷ Characteristics/Problems:

++ well-understood technology (no training need)
– version tracking as a social process (error prone)
– merging diverging versions is annoying (manual process)
– archiving past versions optional/manual (storage problems)
– no multifile support, no snapshots

▷ Summary: only supports serial collaboration, no multifile support



larger teams ⇝ more time wasted

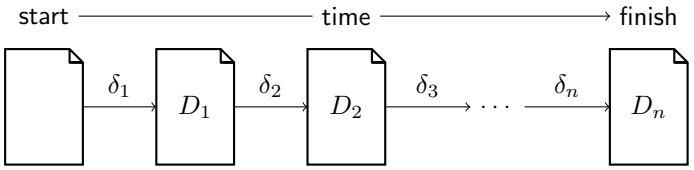
Document Lifecycle Mgmt. & Collaboration Approaches

▷ Practice: Put your documents on Dropbox or MS Sharepoint

▷ Characteristics/Problems:

– local install of (proprietary) software
+ auto-synchronization between cloud and user copies upon save
+ auto-archiving past versions in cloud
– merging diverging versions unsupported (manual process)
– no multifile support, no snapshots

▷ Summary: only supports serial collaboration



larger teams ⇝ more time wasted

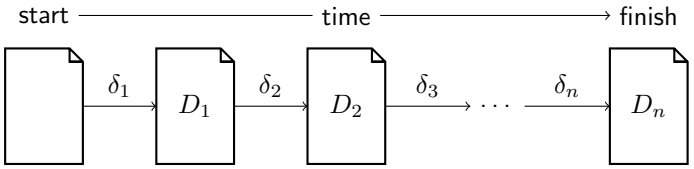
Document Lifecycle Mgmt. & Collaboration Approaches

▷ Practice: Use etherpad, google docs or Office 365 for collaborative editing.

▷ Characteristics/Problems:

+ browser-based, no installation necessary
+ real-time auto-synchronization between cloud and user copies
+ auto-archiving past versions in cloud
+ no diverging versions
– no multifile support, no snapshots

▷ Summary: only supports serial collaboration



larger teams ⇝ more time wasted

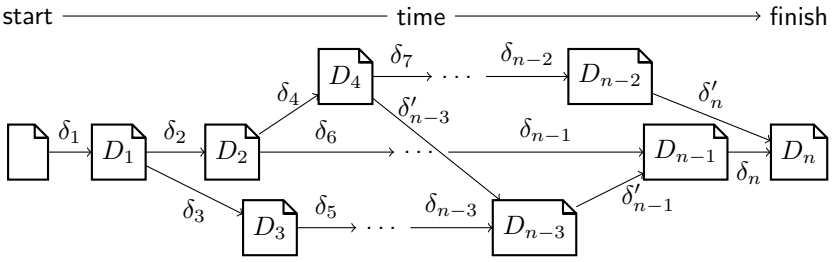
Document Lifecycle Mgmt. & Collaboration Approaches

▷ Practice: Use version control system (for ASCII-based file formats)

▷ Characteristics/Problems:

– special install, training necessary
– restricted to character/line-based formats
+ user-initiated synchronization between cloud and user copies
+ auto-archiving past versions on server
++ multifile support, snapshots, merging support, tagging

▷ Summary: supports parallel, branching collaboration



larger teams ⇝ large-scale parallelization/experimentation

20.2 Centralized Version Control

Centralized version control systems ti

Computing and Managing Differences with diff & patch

▷ **Definition 20.2.1** diff is a file comparison utility that computes differences between two files f_1 and f_2. Differences are output linewise in a diff file (also called a patch), which can be applied to f_1 to obtain f_2 via the patch utility.

▷ **Example 20.2.2**

`The quick brown`
`fox jumps over`
`the lazy dog`
	`The quack brown`
	
	`fox jumps over`
	`the loozy dog`
		`1c1,2`
		`< The quick brown`
		`---`
		`> The quack brown`
		`>`
		`3c4`
		`< the lazy dog`
		`---`
		`> the loozy dog`

▷ **Definition 20.2.3** A diff file consists of a sequence of hunks that in turn consist of a locator which contrasts the source and target locations (in terms of line numbers) followed by the added/deleted lines.

Merging Differences with `merge3`

▷ There are basically two ways of merging the differences of files into one.

▷ **Definition 20.2.4** In two-way merge, an automated procedure tries to combine two different files by copying over differences by guessing or asking the user.

▷ **Definition 20.2.5** In three-way merge the files are assumed to be created by changing a joint original (the parent) by editing. The `merge3` tool examines the differences and patterns appearing in the changes between both files as well as the parent, building a relationship model to generate a new revision. Usually, non-conflicting differences (affecting only one of the files) can directly be copied over.

Definition 20.2.6 A revision control system is a software system that tracks the change process of sets of files via a repository that stores the files' revisions – the content of the files at the time of a commit.

Users do not directly work on the repository, but on a working copy that is synchronized with the repository by revision control actions

- checkout: creates a new working copy from the repository

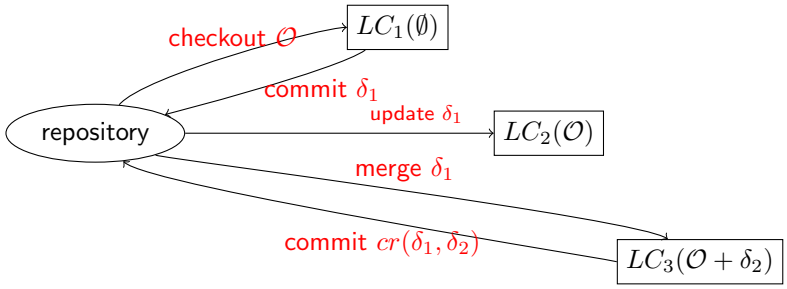
- update: merges the differences between the base revision of the working copy and the revision of the repository into the working copy.
- commit: transmits the differences between the repository revision and the working copy to the repository, which registers them, patches the repository revision, and makes this the new head revision

Version Control with Subversion

▷ **Definition 20.2.7** Subversion is a centralized revision control system that features

- ▷ Central repository (for current revision and reverse diffs)
- ▷ Local working copies (asynchronous checkouts, updates, commits)

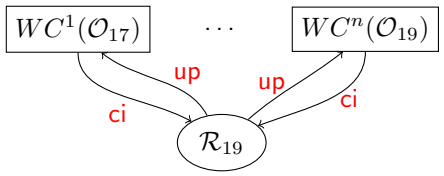
They are kept synchronized by passing around diff differences and patching the repository and working copies. Conflicts are resolved by (three-way) merge.



Collaboration with Subversion

▷ Idea: We can use the same technique for collaboration between multiple working copies.

▷ Diff-Based Collaboration:



The Subversion system takes care of the synchronizeation:

- ▷ you can only commit, if your revision is HEAD (otherwise update)
- ▷ update merges the changes into your working copy
- ▷ If there are changes on the same line, you have a conflict.

```
24 class String
25 <<<<<<< HEAD:lib/jekyll/core_ext.rb
26   def cutoff(desired = 5)
27 =======
28   def cutoff(desired = 400)
29 >>>>>>> conflicts:lib/jekyll/core_ext.rb
30     return self if self.length <= desired
```

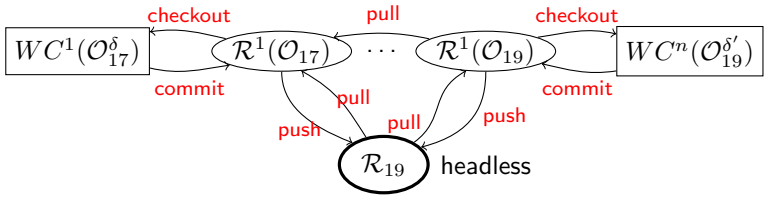

20.3 Distributed Revision Control

Centralized vs. Distributed Version Control

▷ Problem with Subversion:

- ▷ we can only commit when online!
- ▷ all collaboration goes via the repository

▷ Idea: Distribute the Repositories and move differences between them.



Distributed Version Control with git

▷ **Definition 20.3.1** git is a distributed version control system t hat features

- ▷ local repositories (contains head and reverse diffs)
- ▷ local working copies (local commits)
- ▷ multiple remote repositories (branches/forks)
- ▷ local changes can pushed to a remote repository
- ▷ changes from a remote repository can be pulled into the local one.

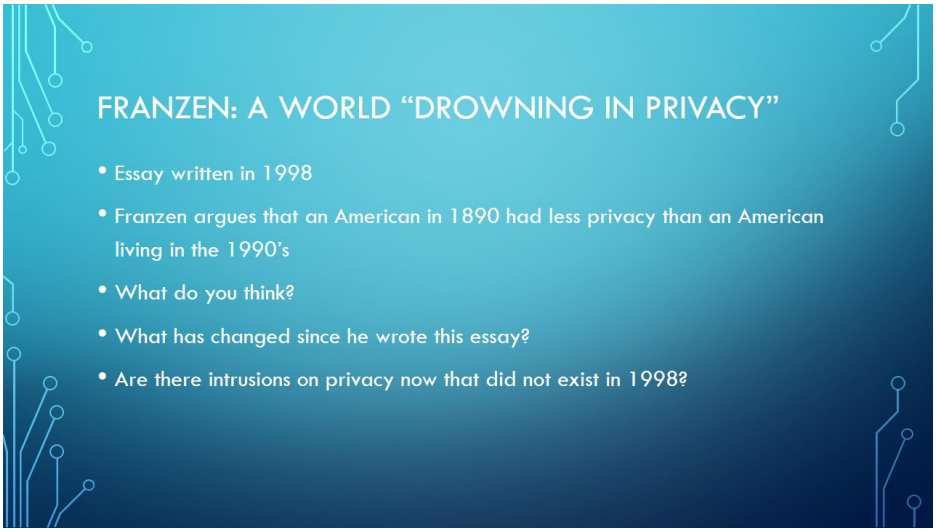
▷ **Definition 20.3.2** There are various repository management systems that facilitate providing repositories, e.g.

- ▷ GitHub, a repository hosting service at `http://GitHub.com`(free public repositories)
- ▷ GitLab, an open source repository management system (http://gitlab.org)

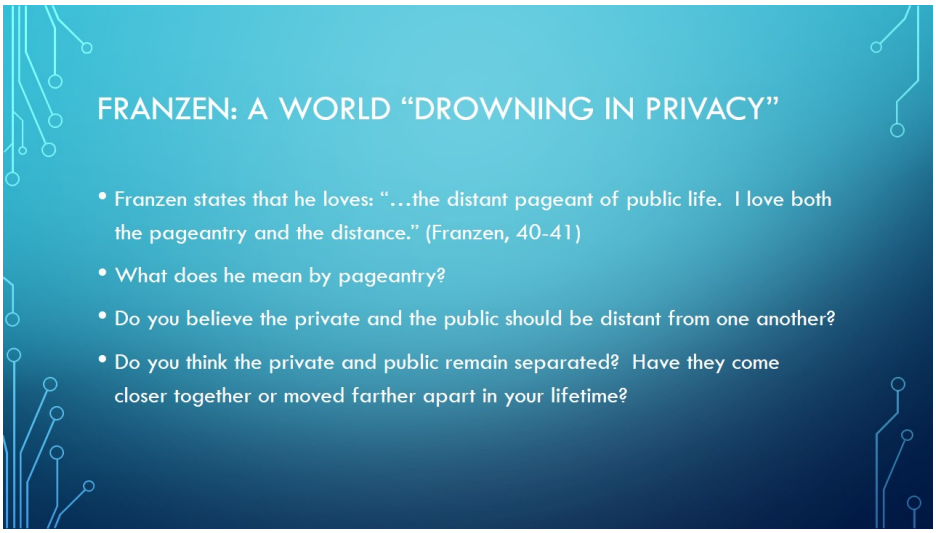
Chapter 21

Privacy, Performance and Identity

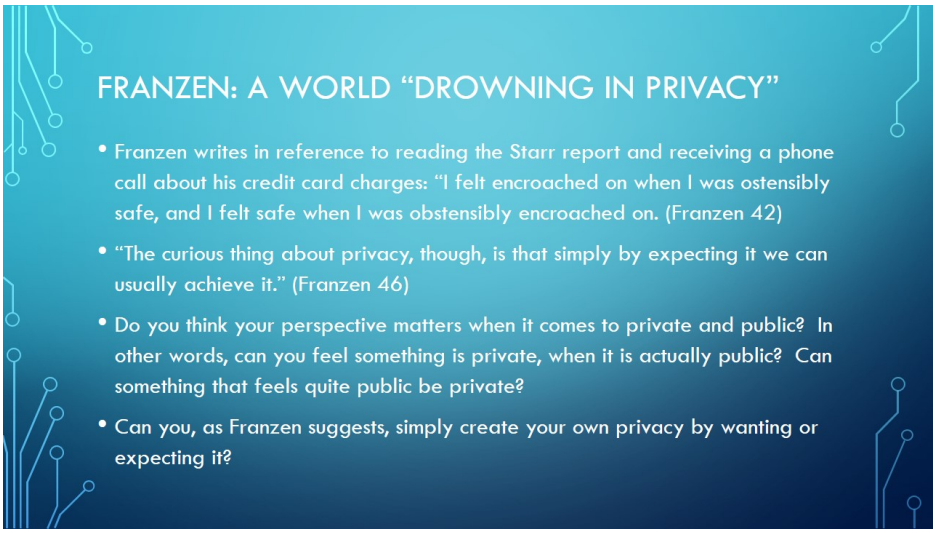
Slide 246



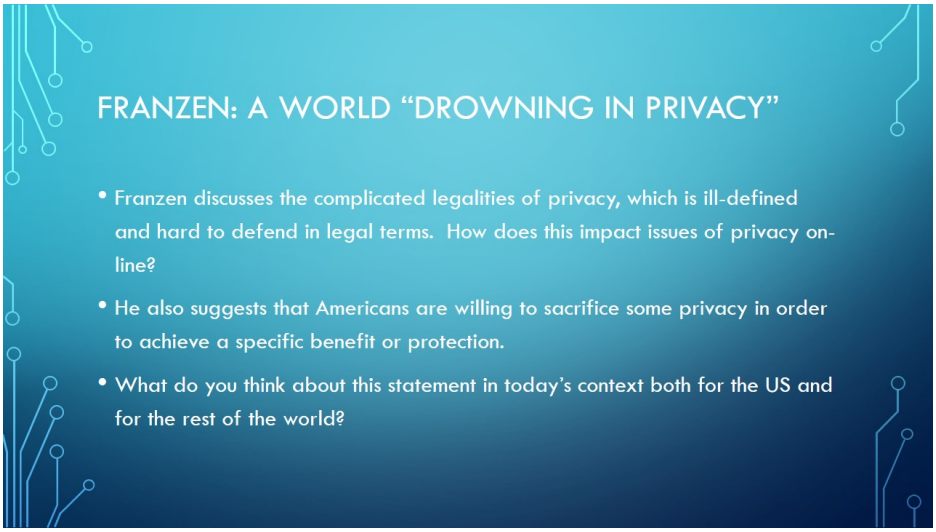
Slide 247



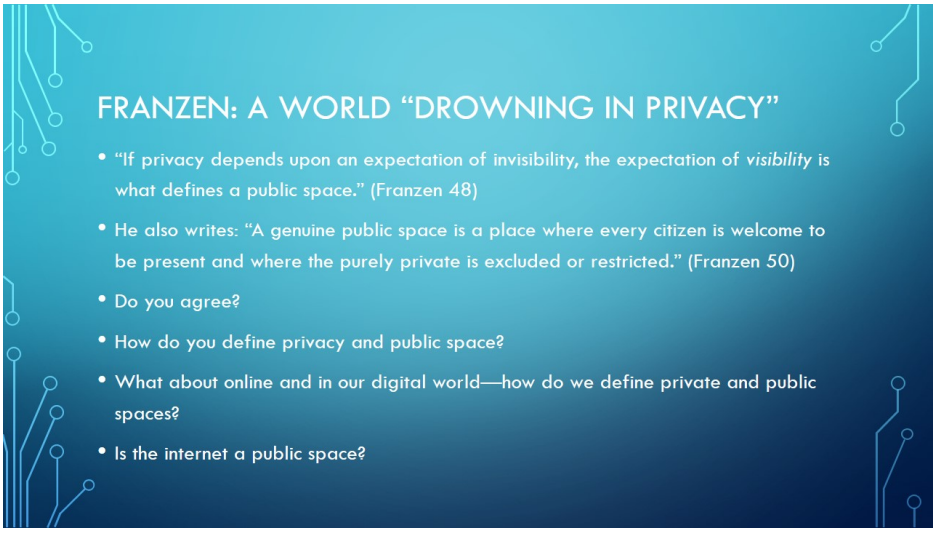
Slide 248



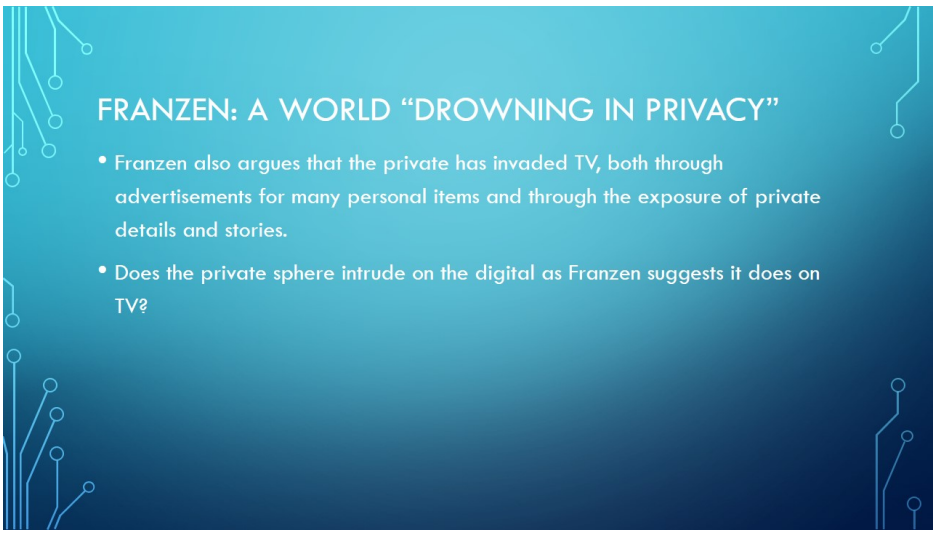
Slide 249



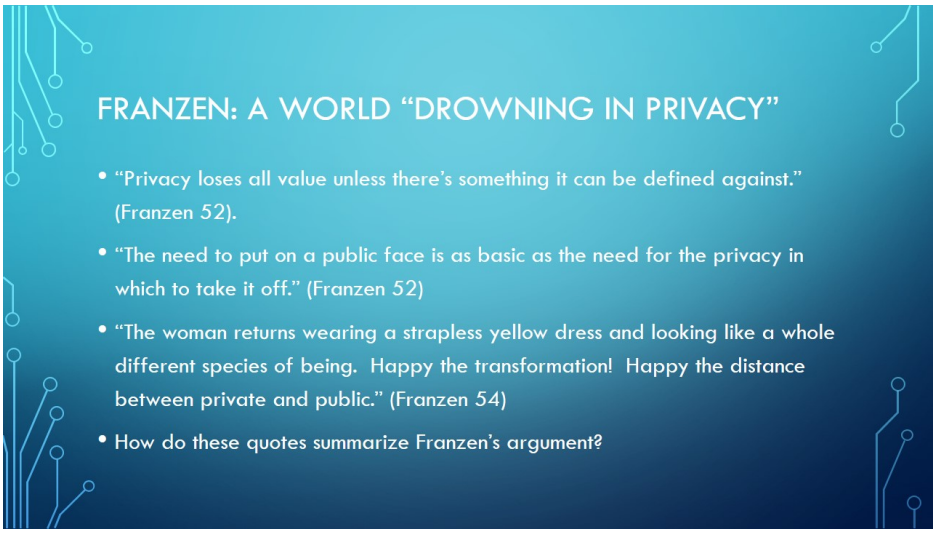
Slide 250



Slide 251

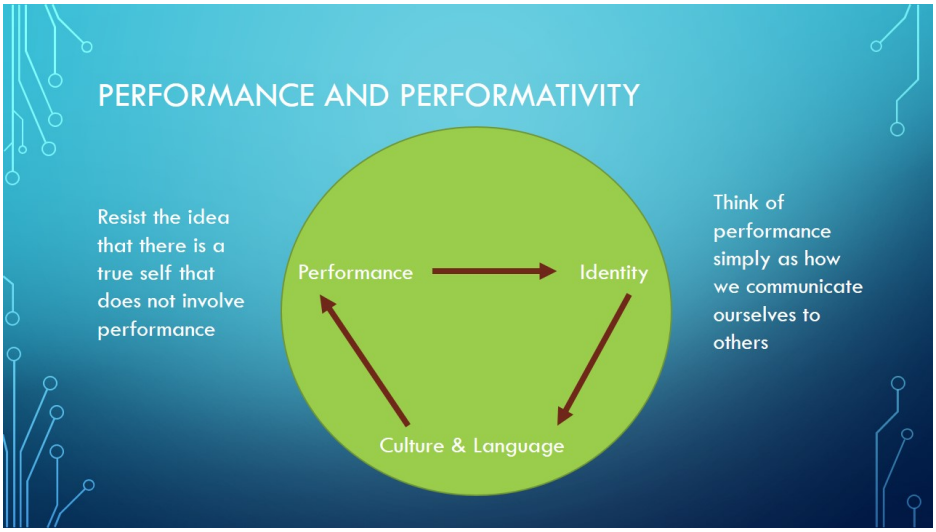


Slide 252

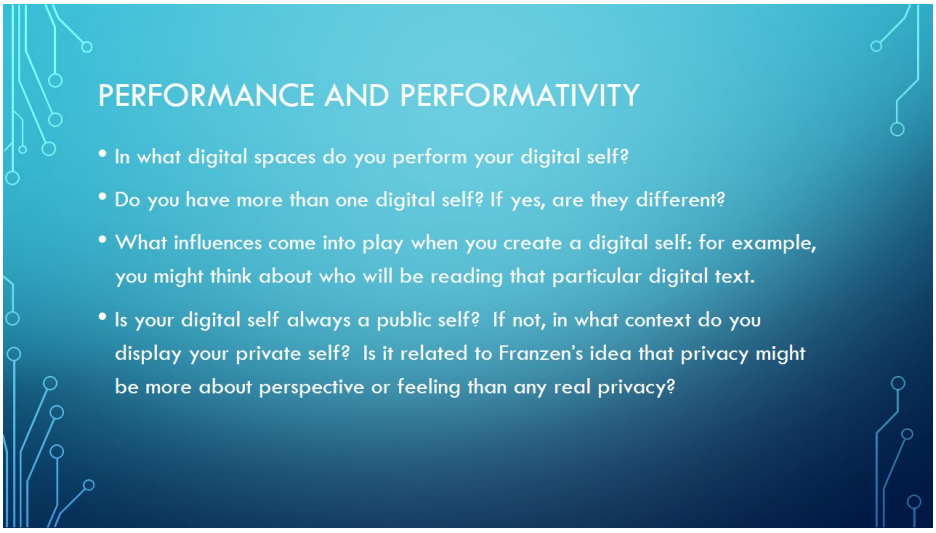


Slide 253

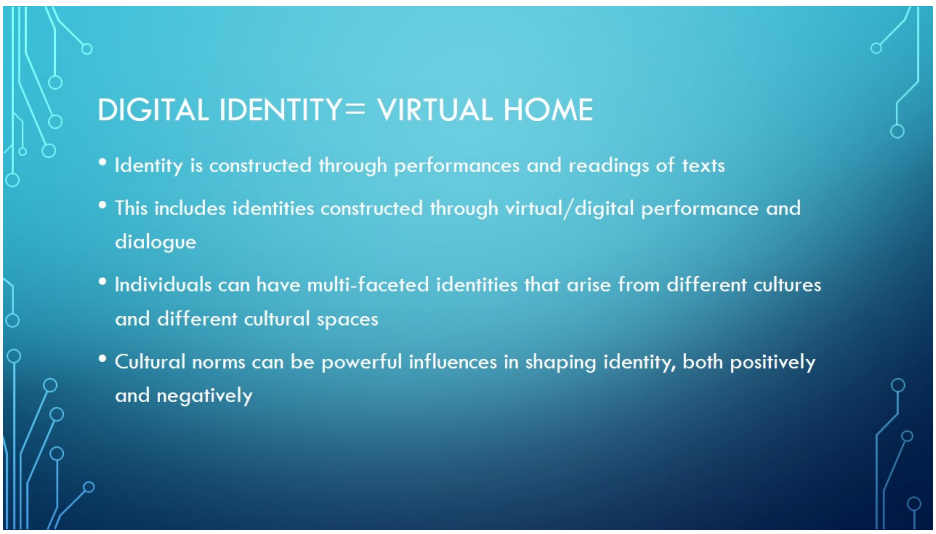
Slide 254



Slide 255

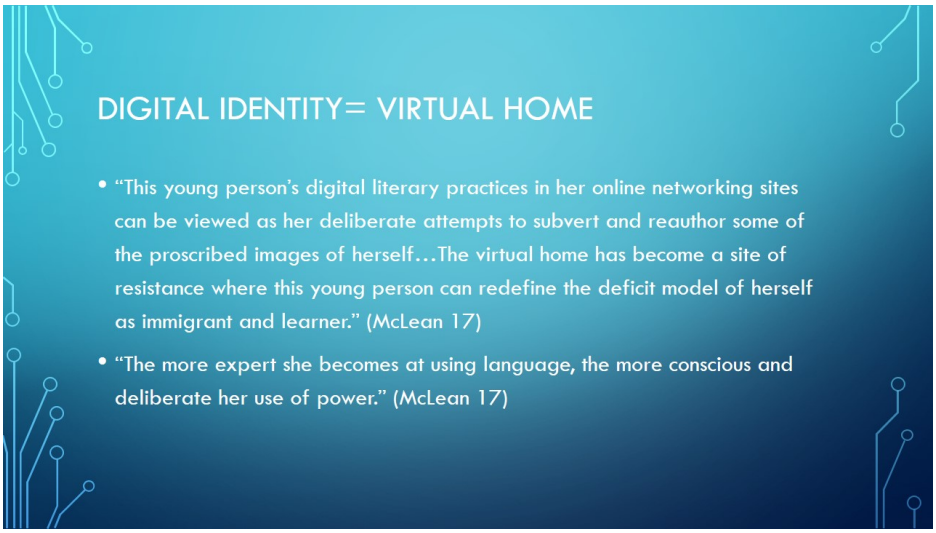


Slide 256



Slide 257

Slide 258



Slide 259

Part III

Intelligent Media and the Future

In this part of the course, we will discuss cutting edge technologies to make digital documets more intelligent, and try to give students a feeling of what such documents and media might bring to our communication behavior and way of living.[9] EdN:9

[9] EdNote: MK: expand

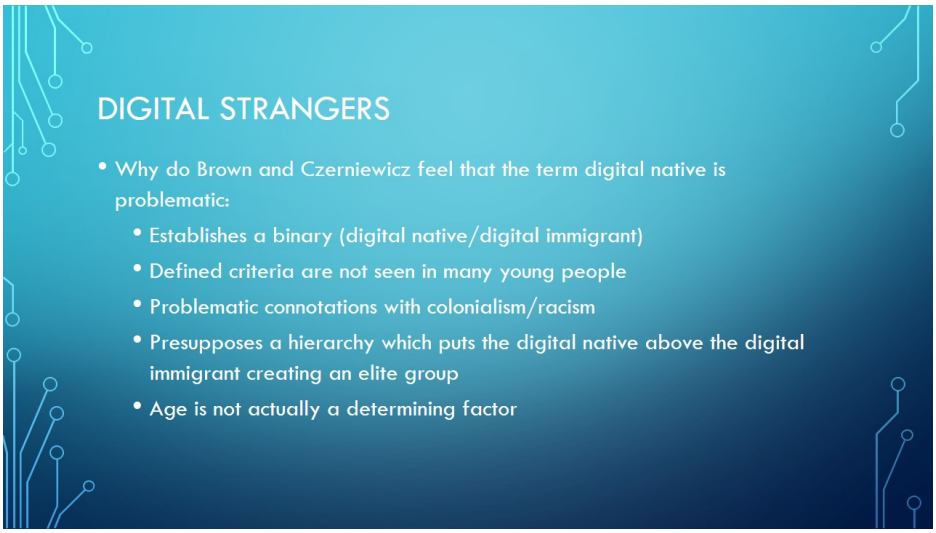
Chapter 22

Digital Generation

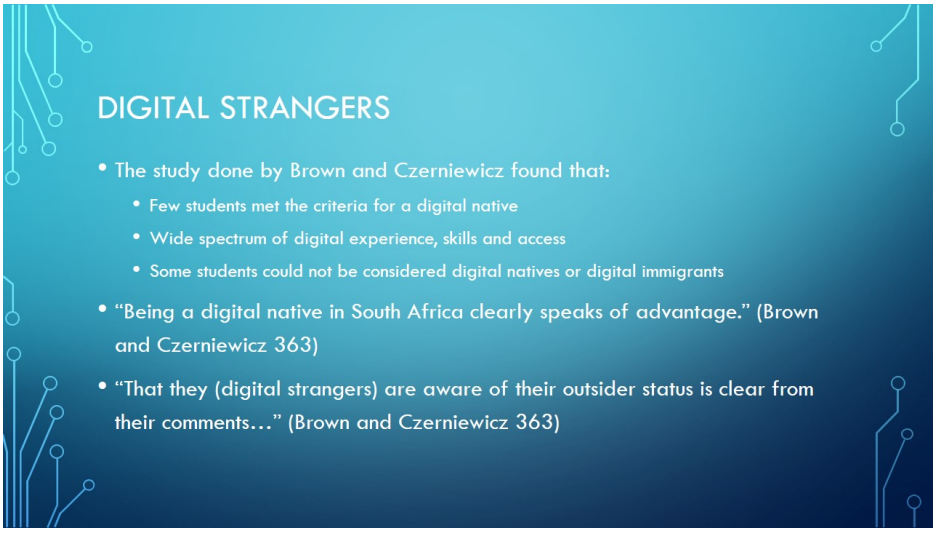
Slide 260

Slide 261

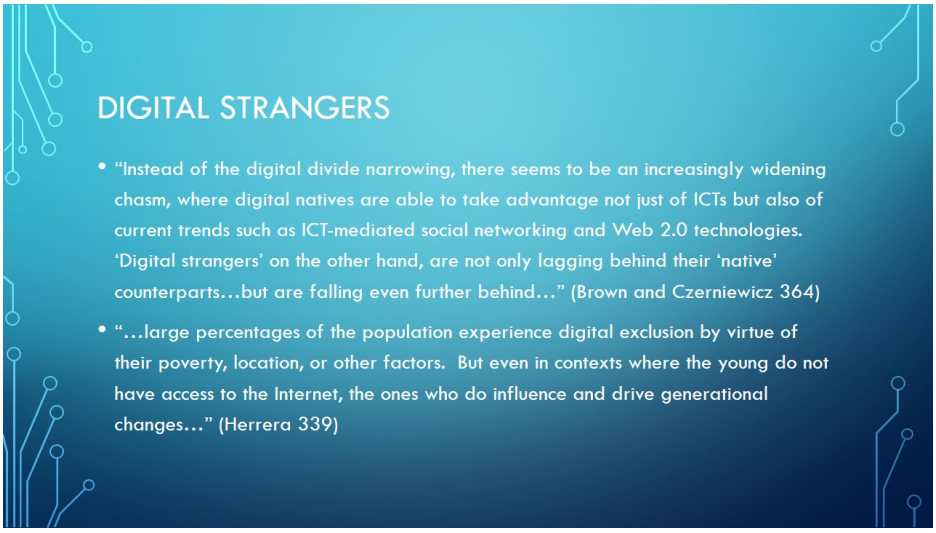
Slide 262



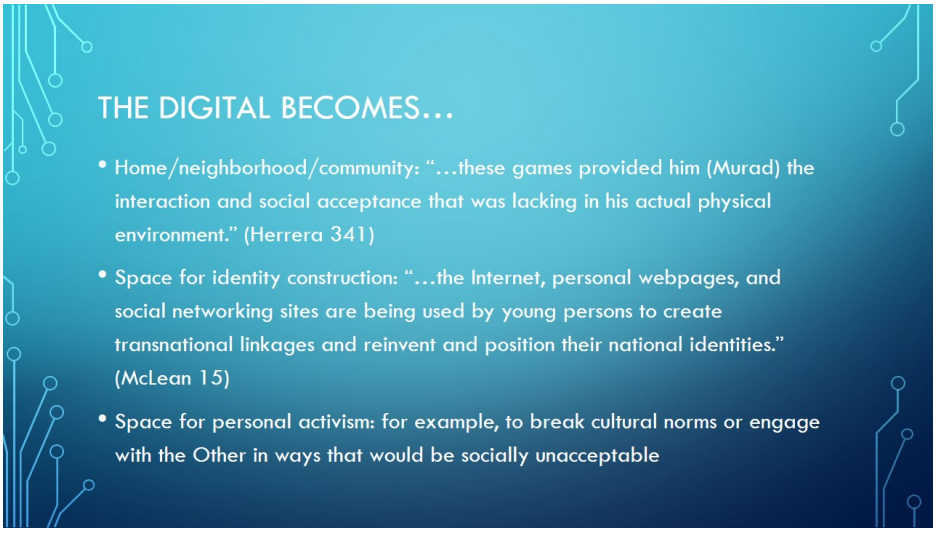
Slide 263



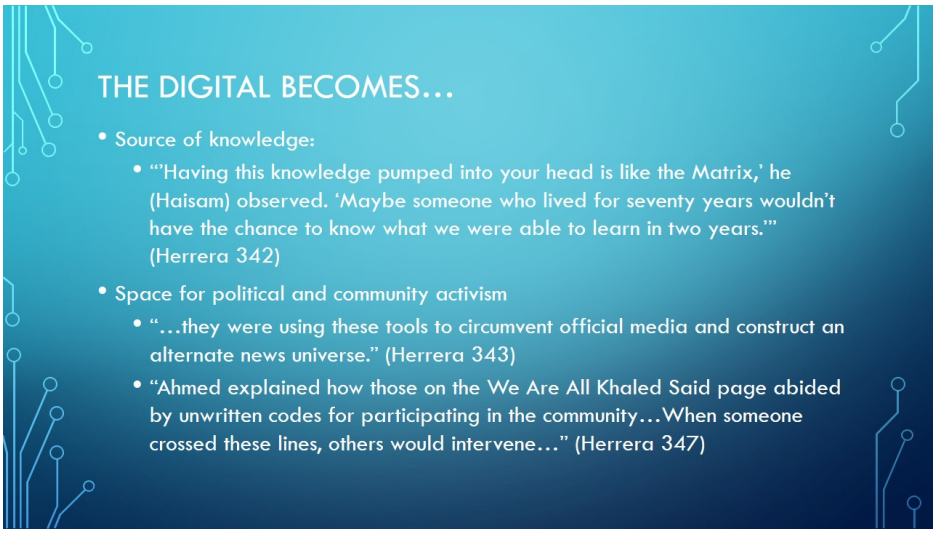
Slide 264



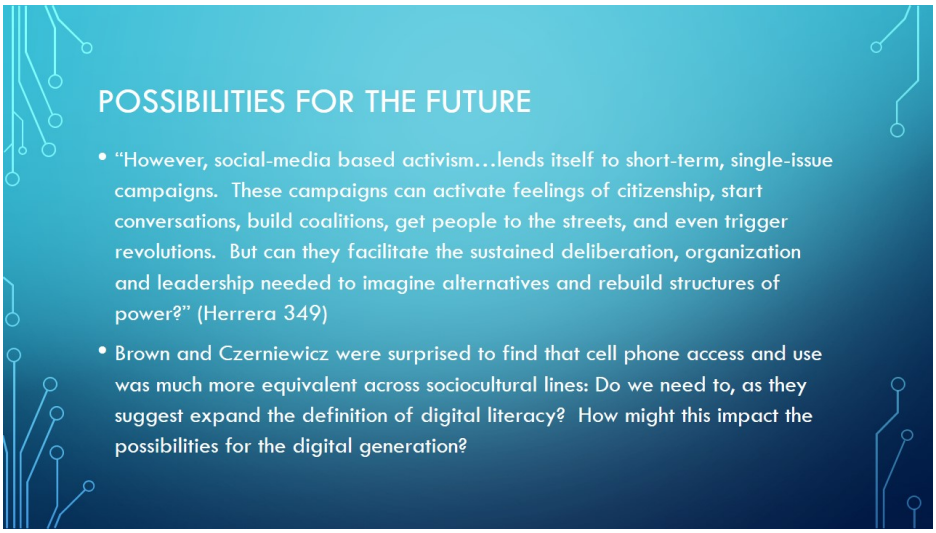
Slide 265



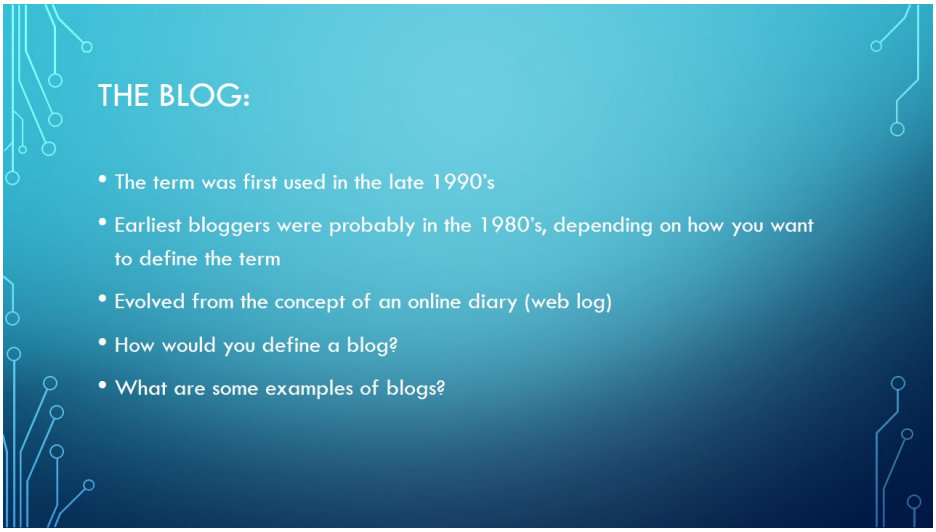
Slide 266



Slide 267



Slide 268



Slide 269

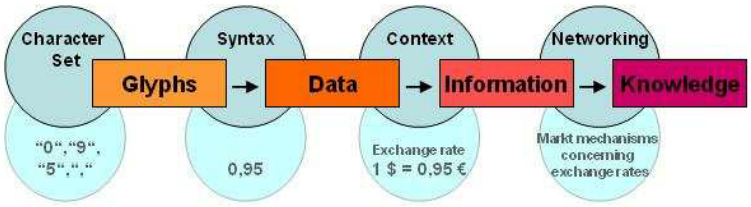
Slide 270

Chapter 23

Knowledge Representation & Semantic Web

What is knowledge? Why Representation?

▷ According to Probst/Raub/Romhardt [PRR97]



For the purposes of this course: Knowledge is the information necessary to support intelligent reasoning

	representation	can be used to determine
	set of words	whether a word is admissible
▷	list of words	the rank of a word
	a lexicon	translation or grammatical function
	structure	function

According to an influential view of [PRR97], knowledge is appears in layers. Staring with a character set that defines a set of glyphs, we can add syntax that turns mere strings into data. Adding context information gives information, and finally, by relating the information to other information allows to draw conclusions, turning information into knowledge.

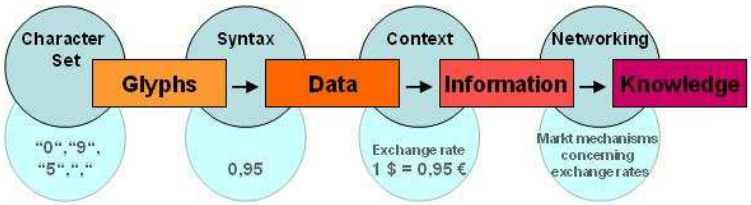
Note that we already have aspects of representation and function in the diagram at the top of the slide. In this, the additional functions added in ¡the successive layers give the representations more and more function, until we reach the knowledge level, where the function is given by inferencing. In the second example, we can see that representations determine possible functions.

23.1 The Semantic Web

The Semantic Web

▷ **Definition 23.1.1** The semantic web is a collaborative movement led by the W3C that promotes the inclusion of semantic content in web pages with the aim of t converting the current web, dominated by unstructured and semi-structured documents into a machine-understandable "web of data".

▷ Idea: Move web content up the ladder, use inference to make connections.



▷ **Example 23.1.2** We want to find information that is not explicitly represented (in one place)

Query: *Who was US president when Barak Obama was born?*

Google: *... BIRTH DATE: August 04, 1961...*

Query: *Who was US president in 1961?*

Google: *President: Dwight D. Eisenhower [...] John F. Kennedy (starting January 20)*

Humans can read (and understand) the text and combine the information to get the answer.

The term "Semantic Web" was coined by Tim Berners Lee in analogy to semantic networks, only applied to the world wide web. And as for semantic networks, where we have inference processes that allow us the recover information that is not explicitly represented from the network (here the world-wide-web).

To see that problems have to be solved, to arrive at the "Semantic Web", we will now look at a concrete example about the "semantics" in web pages. Here is one that looks typical enough.

What is the Information a User sees?

WWW2002
The eleventh International World Wide Web Conference
Sheraton Waikiki Hotel
Honolulu, Hawaii, USA
7-11 May 2002

Registered participants coming from
Australia, Canada, Chile Denmark, France, Germany, Ghana, Hong Kong, India,
Ireland, Italy, Japan, Malta, New Zealand, The Netherlands, Norway,

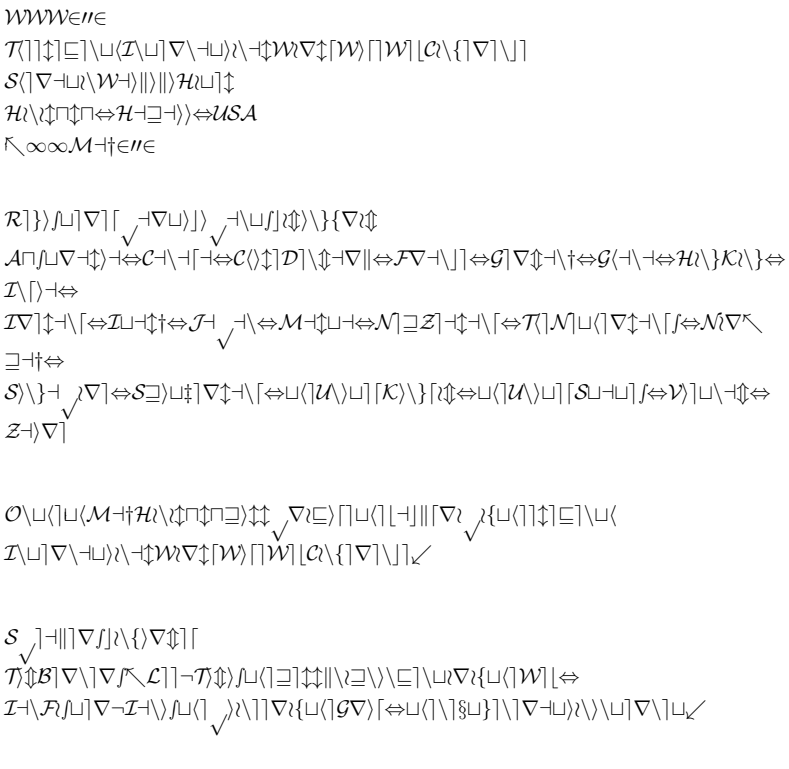
Singapore, Switzerland, the United Kingdom, the United States, Vietnam, Zaire

On the 7th May Honolulu will provide the backdrop of the eleventh
International World Wide Web Conference.

Speakers confirmed
Tim Berners-Lee: Tim is the well known inventor of the Web,
Ian Foster: Ian is the pioneer of the Grid, the next generation internet.

But as for semantic networks, what you as a human can see ("understand" really) is deceptive, so let us obfuscate the document to confuse your "semantic processor". This gives an impression of what the computer "sees".

What the machine sees



Obviously, there is not much the computer understands, and as a consequence, there is not a lot the computer can support the reader with. So we have to "help" the computer by providing some meaning. Conventional wisdom is that we add some semantic/functional markup. Here we pick XML without loss of generality, and characterize some fragments of text e.g. as dates.

Solution: XML markup with "meaningful" Tags

<title>WWW∈//∈
T⌉]↕]⊑]\⊔⟨I\⊔]∇\⊣⊔⟩ι\⊣↕W∇↕⌈W⟩⌈W⌉⌊C∪\{⌉∇⌉\⌋]</title>
<place>S⟨⌉∇⊣⊔ι\W⊣⟩‖⟩‖⟩H∪⊔]↕Hι\ι↕⊓↕⊓⇔H⊣⊒⊣⟩⟩⇔USA</place>
<date>↖∞∞M⊣†∈//∈</date>
<participants>R]}⟩∫⊔]∇⌉⌈ √¬∇⊔⟩⌋⟩ √⊣\⊔∫]ι⇳⟩\}{∇ι⇳
A⊓∫⊔∇⊣⇳⟩⊣⇔C⊣\⊣⌈⊣⇔C⟨⟩↕]D]\⇳⊣∇‖⇔F∇⊣\⌋]⇔G]∇⇳⊣\†⇔G⟨⊣\⊣⇔Hι\}Kι\}⇔
I\⌈⟩⊣⇔
I∇]↕⊣\⌈⇔I⊔⊣⇳†⇔J⊣ √⊣\⇔M⊣⇳⊔⊣⇔N]⊒Z]⊣⇳⊣\⌈⇔T⌉]N]⊔⟨⌉∇↕⊣\⌈∫⇔Nι∇↖
⊒⊣†⇔
S⟩\}⊣ √∇]⇔S⊒⟩⊔‡]∇↕⊣\⌈⇔⊔⟨⌉U\⟩⊔]⌈K⟩\}⌈ι⇳⇔⊔⟨⌉U\⟩⊔]⌈S⊔⊣⊔]∫⇔V⟩]⊔\⊣⇳⇔
Z⊣⟩∇]</participants>
</introduction>O\⊔⟨⌉⊔⟨M⊣†Hι\ι⇳⊓↕⊓⊒⟩⇳ √∇ι⊑⟩⌈]⊔⟨⌉⌊⊣⌋‖⌈∇ι √ι{⊔⟨⌉]↕]⊑]\⊔⟨
I\⊔]∇\⊣⊔⟩ι\⊣↕W∇↕⌈W⟩⌈W⌉⌊C∪\{⌉∇⌉\⌋]↙</introduction>
<program>S √]⊣‖]∇∫⌋ι\{⟩∇⇳]⌈
<speaker>T⟩⇳B]∇\]∇∫↖L]]¬T⟩⇳⟩∫⊔⟨⌉⊒]↕⇳‖\ι⊒\⟩\⊑]\⊔ι∇ι{⊔⟨⌉W⌉⌊</speaker>
<speaker>I⊣\FιJ⊔]∇¬I⊣\⟩∫⊔⟨⌉ √⟩ι\⌉]∇ι{⊔⟨⌉G∇⟩⌈⇔⊔⟨⌉\⌉§⊔}]\⌉∇⊣⊔⟩ι\⟩\↖
⊔]∇\]⊔<speaker>
</program>

What can we do with this?

▷ **Example 23.1.3** Consider the following fragments:

<title>WWW∈//∈
T⌉]↕]⊑]\⊔⟨I\⊔]∇\⊣⊔⟩ι\⊣↕W∇↕⌈W⟩⌈W⌉⌊C∪\{⌉∇⌉\⌋]</title>
<place>S⟨⌉∇⊣⊔ι\W⊣⟩‖⟩‖⟩H∪⊔]↕Hι\ι⇳⊓↕⊓⇔H⊣⊒⊣⟩⟩⇔USA</place>
<date>↖∞∞M⊣†∈//∈</date>

Given the markup above, we can

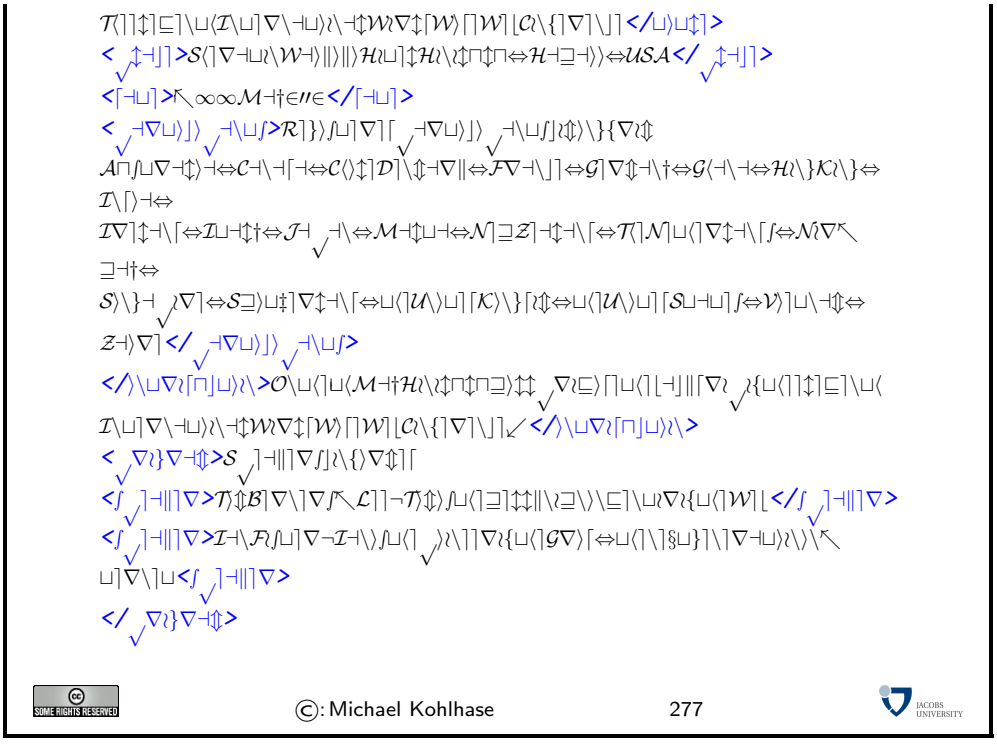
- ▷ parse ↖∞∞M⊣†∈//∈ as the date May 7-11 2002 and add this to the user's calendar.
- ▷ parse S⟨⌉∇⊣⊔ι\W⊣⟩‖⟩‖⟩H∪⊔]↕Hι\ι⇳⊓↕⊓⇔H⊣⊒⊣⟩⟩⇔USA as a destination and find flights.

But: do not be deceived by your ability to understand English

We have to obfuscate the markup as well, since it does not carry any meaning to the machine intrinsically either.

▷ What the machine sees of the XML

<⊔⟩⊔↕]>WWW∈//∈

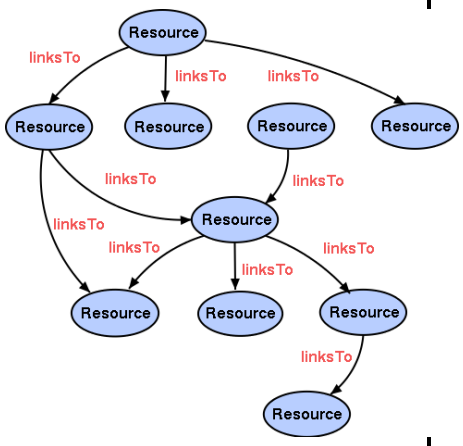


So we have not really gained much either with the markup, we really have to give meaning to the markup as well, this is where techniques from knowledge representation come into play

To understand how we can make the web more semantic, let us first take stock of the current status of (markup on) the web. It is well-known that world-wide-web is a hypertext, where multimedia documents (text, images, videos, etc. and their fragments) are connected by hyperlinks. As we have seen, all of these are largely opaque (non-understandable), so we end up with the following situation (from the viewpoint of a machine).

The Current Web

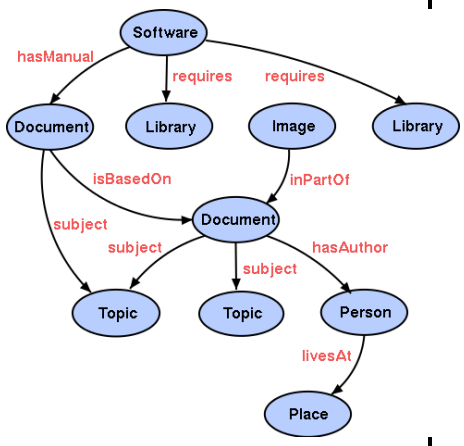
- ▷ Resources: identified by URI's, untyped
- ▷ Links: `href`, `src`, ... limited, non-descriptive
- ▷ User: Exciting world - semantics of the resource, however, gleaned from content
- ▷ Machine: Very little information available - significance of the links only evident from the context around the anchor.



Let us now contrast this with the envisioned semantic web.

The Semantic Web

- ▷ Resources: Globally Identified by URI's or Locally scoped (Blank), Extensible, Relational
- ▷ Links: Identified by URI's, Extensible, Relational
- ▷ User: Even more exciting world, richer user experience
- ▷ Machine: More processable information is available (Data Web)
- ▷ Computers and people: Work, learn and exchange knowledge effectively



Essentially, to make the web more machine-processable, we need to classify the resources by the concepts they represent and give the links a meaning in a way, that we can do inference with that.

The ideas presented here gave rise to a set of technologies jointly called the "semantic web", which we will now summarize before we return to our logical investigations of knowledge representation techiques.

Need to add "Semantics"

- ▷ External agreement on meaning of annotations E.g., Dublin Core
  - ▷ Agree on the meaning of a set of annotation tags
  - ▷ Problems with this approach: Inflexible, Limited number of things can be expressed
- ▷ Use Ontologies to specify meaning of annotations
  - ▷ Ontologies provide a vocabulary of terms
  - ▷ New terms can be formed by combining existing ones
  - ▷ Meaning (semantics) of such terms is formally specified
  - ▷ Can also specify relationships between terms in multiple ontologies
- ▷ Inference with annotations and ontologies (get out more than you put in!)
  - ▷ Standardize annotations in RDF [KC04] or RDFa [HASB13] and ontologies on OWL [OWL09]
  - ▷ Harvest RDF and RDFa in to a triplestore or OWL reasoner.
  - ▷ Query that for implied knowledge(e.g. chaining multiple facts from Wikipedia)

    SPARQL: Who was US President when Barack Obama was Born?
    DBPedia: John F. Kennedy (was president in August 1961)

23.2 Semantic Networks

To get a feeling for early knowledge representation approaches from which description logics developed, we take a look at "semantic networks" and contrast them to logical approaches.

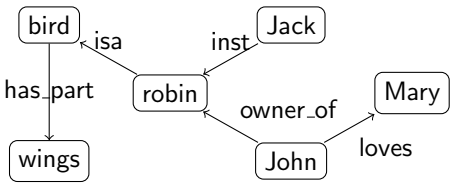
Semantic networks are a very simple way of arranging concepts and their relations in a graph.

Semantic Networks [CQ69]

▷ **Definition 23.2.1** A semantic network is a graph structure for representing knowledge:

- ▷ nodes represent concepts (e.g. bird, John, robin)
- ▷ links represent relations between these (isa, father_of, belongs_to)

▷ **Example 23.2.2** A semantic net for birds and persons:



Problem: how do we do inference from such a network?

▷▷ Idea: encode taxonomic information about concepts and individuals

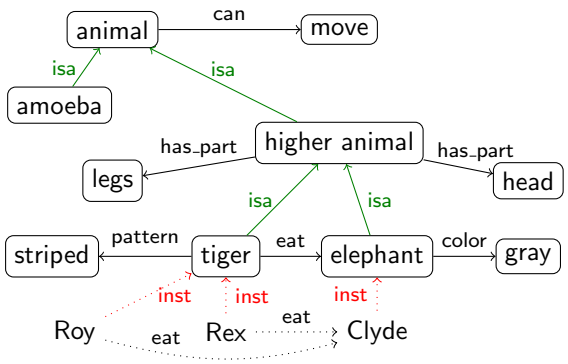
- ▷ in "isa" links (inclusion of concepts)
- ▷ in "inst" links (concept memberships)
- ▷ use property inheritance along "isa" and "inst" in the process model

Even though the network in Example 23.2.2 is very intuitive (we immediately understand the concepts depicted), it is unclear how we (and more importantly a machine that does not associate meaning with the labels of the nodes and edges) can draw inferences from the "knowledge" represented.

Another problem is that the semantic net in Example 23.2.2 confuses two kinds of concepts: individuals (represented by proper names like *John* and *Jack*) and concepts (nouns like *robin* and *bird*). Even though the "isa" and "inst" links already acknowledge this distinction, the "has_part" and "loves" relations are at different levels entirely, but not distinguished in the networks.

Terminologies and Assertions

▷ **Example 23.2.3** From the network



infer that *elephants* have *legs* and that *Clyde* is *gray*.

▷ **Definition 23.2.4** We call the subgraph of a semantic network N spanned by the "isa" relations the terminology (or TBox, or the famous Isa-Hierarchy) and the subgraph spanned by the "inst" relation the assertions (or ABox) of N.

But there are sever shortcomings of semantic networks: the suggestive shape and node names give (humans) a false sense of meaning, and the inference rules are only given in the process model (the implementation of the semantic network processing system).

This makes it very difficult to assess the strength of the inference system and make assertions e.g. about completeness.

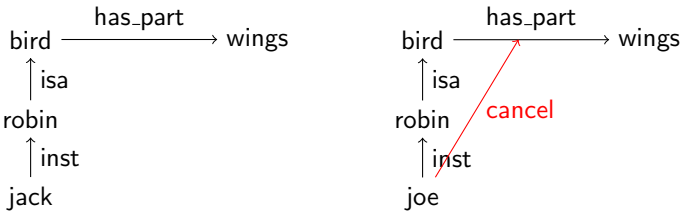
Limitations of Semantic Networks

▷ What is the meaning of a link?

- ▷ link names are very suggestive (misleading for humans)
- ▷ meaning of link types defined in the process model(no denotational semantics)

Problem: No distinction of optional and defining traits

▷▷ **Example 23.2.5** Consider a robin that has lost its wings in an accident



Cancel-links have been proposed, but their status and process model are debatable.

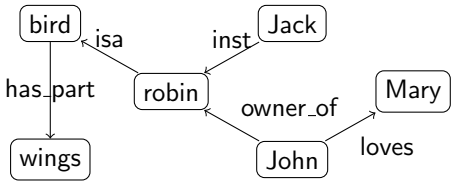
To alleviate the perceived drawbacks of semantic networks, we can contemplate another notation that is more linear and thus more easily implemented: function/argument notation.

Another Notation for Semantic Networks

▷ Idea: use function/argument notation

- ▹ Interpret nodes as arguments (reification to individuals)
- ▹ Interpret links as functions (logical relations)

▷ **Example 23.2.6**



isa(robin,bird)
haspart(bird,wings)
inst(Jack,robin)
owner_of(John, robin)
loves(John,Mary)

▷ Evaluation:

- \+ linear notation (equivalent, but better to implement on a computer)
- \+ easy to give process model by deduction (e.g. in ProLog)
- – worse locality properties (networks are associative)

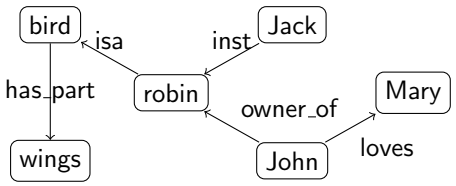
Indeed the function/argument notation is the immediate idea how one would naturally represent semantic networks for implementation.

This notation has been also characterized as subject/predicate/object triples, alluding to simple (English) sentences. This will play a role in the "semantic web" later.

Building on the function/argument notation from above, we can now give a formal semantics for semantic networks: we translate into first-order logic and use the semantics of that.

A Denotational Semantics for Semantic Networks

▷ Extension: take isa/inst concept/individual distinction into account



$robin \subseteq bird$
haspart(bird,wings)
$Jack \in robin$
owner_of(John, Jack)
loves(John,Mary)

▷ Observation: this looks like first-order logic, if we take

- ▹ $A \subseteq B$ to mean $\forall X.A(X) \Rightarrow B(X)$
- ▹ $a \in S$ to mean $S(a)$
- ▹ $haspart(A,B)$ to mean $\forall X.A(X) \Rightarrow (\exists Y.B(Y) \wedge part\_of(X,Y))$

▷ Idea: Take first-order deduction as process model(gives inheritance for free)

Indeed, the semantics induced by the translation to first-order logic, gives the intuitive meaning to the semantic networks. Note that this only holds only for the features of semantic networks

that are representable in this way, e.g. the cancel links shown above are not (and that is a feature, not a bug).

But even more importantly, the translation to first-order logic gives a first process model: we can use first-order inference to compute the set of inferences that can be drawn from a semantic network.

23.3 Description Logics and the Semantic Web

Resource Description Framework

▷ **Definition 23.3.1** The Resource Description Framework (RDF) is a framework for describing resources on the web. It is a XML vocabulary developed by the W3C.

▷ Note: RDF is designed to be read and understood by computers, not to be being displayed to people

▷ **Example 23.3.2** RDF can be used for describing

- ▷ properties for shopping items, such as price and availability
- ▷ time schedules for web events
- ▷ information about web pages (content, author, created and modified date)
- ▷ content and rating for web pictures
- ▷ content for search engines
- ▷ electronic libraries

Resources and URIs

▷ RDF describes resources with properties and property values.

▷ RDF uses Web identifiers (URIs) to identify resources.

▷ **Definition 23.3.3** A resource is anything that can have a URI, such as `http://www.jacobs-university.de`

▷ **Definition 23.3.4** A property is a resource that has a name, such as *author* or *homepage*, and a property value is the value of a property, such as *Michael Kohlhase* or `http://kwarc.info/kohlhase`(a property value can be another resource)

▷ **Definition 23.3.5** The combination of a resource, a property, and a property value forms a statement (known as the subject, predicate and object of a statement).

▷ **Example 23.3.6** Statement: *The [author]^{pred} of [this slide]^{subj} is [Michael Kohlhase]^{obj}*

XML Syntax for RDF

- RDF is a concrete XML vocabulary for writing statements
- **Example 23.3.7** The following RDF document could describe the slides as a resource

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc= "http://purl.org/dc/elements/1.1/">
  <rdf:Description about="https://svn.kwarc.info/.../CompLog/kr/en/rdf.tex">
    <dc:creator>Michael Kohlhase</dc:creator>
    <dc:source>http://www.w3schools.com/rdf</dc:source>
  </rdf:Description>
</rdf:RDF>
```

This RDF document makes two statements:

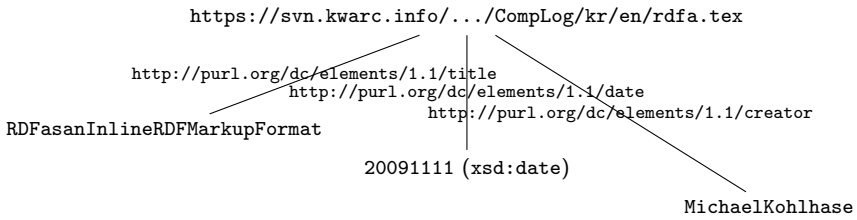
- The subject of both is given in the `about` attribute of the `rdf:Description` element
- The predicates are given by the element names of its children
- The objects are given in the elements as URIs or literal content.

Intuitively: RDF is a way to write down ABox information in a web-scalable way.

RDFa as an Inline RDF Markup Format

- Problem: RDF is a standoff markup format(annotate by URIs pointing into other files)
- **Example 23.3.8**

```
<div xmlns:dc="http://purl.org/dc/elements/1.1/">
    <h2 property="dc:title">RDF as an Inline RDF Markup Format</h2>
    <h3 property="dc:creator">Michael Kohlhase</h3>
    <em property="dc:date" datatype="xsd:date"
       content="20091111">November 11., 2009</em>
</div>
```



OWL as an Ontology Language for the Semantic Web

▷ Idea: Use Description Logics to talk about RDF triples.

▷ An RDF triple is an ABox entry for a role constraint $h\mathrm{R}s$

▷ **Example 23.3.9** h is the resource for Ian Horrocks, s is the resource for Ulrike Sattler, and R is the the relation "hasColleague" in

```
<rdf:Description about="some.uri/person/ian_horrocks">
  <hasColleague resource="some.uri/person/uli_sattler"/>
</rdf:Description>
```

Idea: Now collect similar resources in *classes*, and state rules about them in a way, so that we can use inference to make knowledge explicit that was implicit before (saves us lots of work!)

▷▷ Idea: We know how to do this, this is just $\mathcal{ALC}$+!!!

The OWL Language

▷ Three species of OWL

- ▷ OWL Full is union of OWL syntax and RDF
- ▷ OWL DL restricted to FOL fragment
- ▷ OWL Lite is "easier to implement" subset of OWL DL

▷ Semantic layering

- ▷ OWL DL $\hat{=}$ OWL Full within DL fragment
- ▷ DL semantics officially definitive
- ▷ OWL DL based on SHIQ Description Logic($\mathcal{ALC}$ + number restrictions, transitive roles, inverse roles, role inclu
- ▷ OWL DL benefits from many years of DL research
- ▷ Well defined semantics, formal properties well understood (complexity, decidability)
- ▷ Known reasoning algorithms, Implemented systems (highly optimized)

Chapter 24

MathML: Content vs. Presentation Markup

24.1 MathML: Presentation and Content of Mathematical Formulae

Representation of Formulae as Expression Trees

▷ Mathematical Expressions are build up as expression trees

- ▷ of layout schemata in Presentation-MathML
- ▷ of functional subexpressions in Content-MathML

▷ Example: $\frac{3}{x+2}$

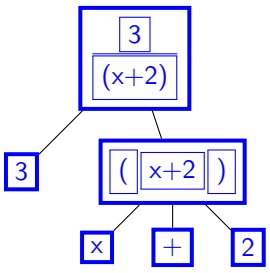
```
<mfrac>
  <mn>3</mn>
  <mfenced>
     <mi>x</mi>
     <mo>+</mo>
     <mn>2</mn>
   </mfenced>
</mfrac>
```

```
<apply>
  <divide/>
  <cn>3</cn>
  <apply>
    <plus/>
    <ci>x</ci>
    <cn>2</cn>
  </apply>
</apply>
```


Layout Schemata and the MathML Box model

▷ Presentation MathML represents the visual appearance of a formula in a tree of layout primitives

▷ **Example 24.1.1 (Presentation MathML for $3/(x+2)$)**

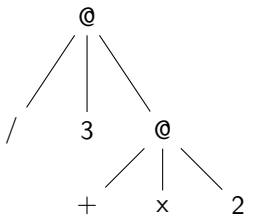



```
<mfrac>...</mfrac>
<mn>3</mn>
<mfenced>...</mfenced>
<mi>x</mi>
<mo>+</mo>
<mn>2</mn>
```


Functional Markup in MathML: The "Operator Tree"

▷ Content MathML represents the functional structureof a formula in a tree of operators, via application and binding.

▷ **Example 24.1.2 (Content MathML for $3/(x+2)$)**




```
<apply>...</apply>
<div/>
<cn>3</cn>
<apply>...</apply>
<plus/>
<ci>x</ci>
<mn>2</mn>
```

Extra Operators: use `<csymbol cd="⟪CD⟫">⟪Name⟫</csymbol>`, where

▷ ▷ ⟪CD⟫ is a content dictionary – a document that defines ⟪Name⟫

▷ ⟪Name⟫ is the name of a symbol definition in ⟪CD⟫.

Content Mathml: Expression Trees in Prefix Notation

▷ Prefix Notation saves parentheses (so does postfix, BTW)

$(x-y)/2$	$x-(y/2)$

`<apply>`
`  <divide/>`
`  <apply>`
`  <minus/>`
`  <ci>x</ci>`
`  <ci>y</ci>`
`  </apply>`
`  <cn>2</cn>`
`</apply>`
	`<apply>`
	`  <minus/>`
	`  <ci>x</ci>`
	`  <apply>`
	`  <divide/>`
	`  <ci>y</ci>`
	`  <cn>2</cn>`
	`  </apply>`
	`</apply>`

Function Application: `<apply>function arg1 ... argn </apply>`

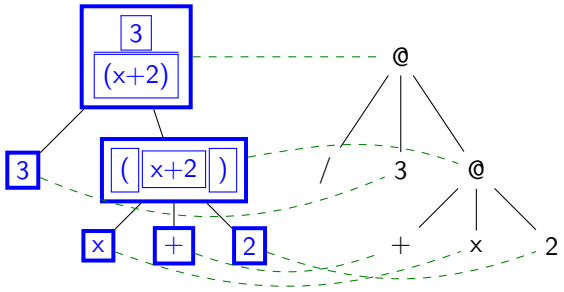
▷▷ Operators and Functions: ∼ 100 empty elements `<sin/>`, `<plus/>`, `<eq/>`, `<compose/>`,...

▷ Token elements: `ci`, `cn` (identifiers and numbers)

▷ Extra Operators: `<csymbol cd="...">...</csymbol>`

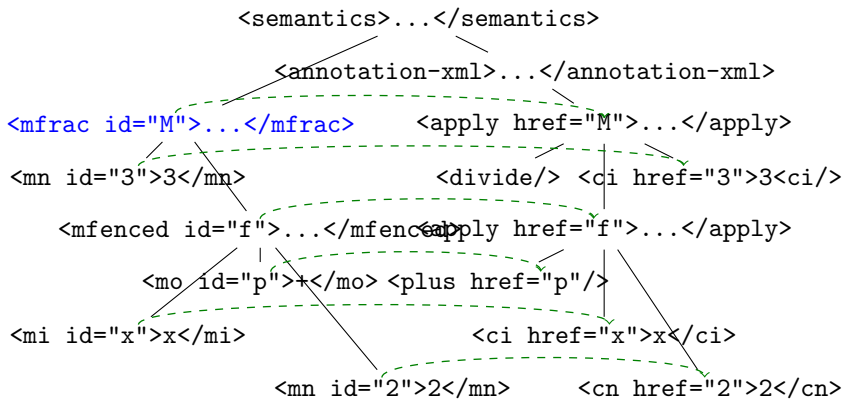
Parallel Markup e.g. in MathML

▷ Idea: Combine the presentation and content markup and cross-reference



▷ use e.g. for semantic copy and paste.(click on presentation, follow link and copy content)

▷ Concrete Realization in MathML: `semantics` element with presentation as first child and content in `annotation-xml` child



Mixing Presentation and Content MathML

```
<semantics>
 <mrow>
  <mrow><mo>(</mo><mi>a</mi> <mo>+</mo> <mi>b</mi><mo>)</mo></mrow>
  <mo>&InvisibleTimes;</mo>
```

```
 <mrow><mo>(</mo><mi>c</mi> <mo>+</mo> <mi>d</mi><mo>)</mo></mrow>
</mrow>
<annotation-xml encoding="MathML-Content">
 <apply><times/>
  <apply><plus/><ci>a</ci> <ci>b</ci></apply>
  <apply><plus/><ci>c</ci> <ci>d</ci></apply>
 </apply>
</annotation-xml>
<annotation-xml encoding="openmath">
 <OMA><OMS cd="arithmetics" name="times"/>
  <OMA><OMS cd="arithmetics" name="plus"/><OMV name="a"/><OMV name="b"/></OMA>
  <OMA><OMS cd="arithmetics" name="plus"/><OMV name="c"/><OMV name="d"/></OMA>
 </OMA>
</annotation-xml>
</semantics>
```


24.2 Presentation MathML

Representation of Formulae as Expression Trees

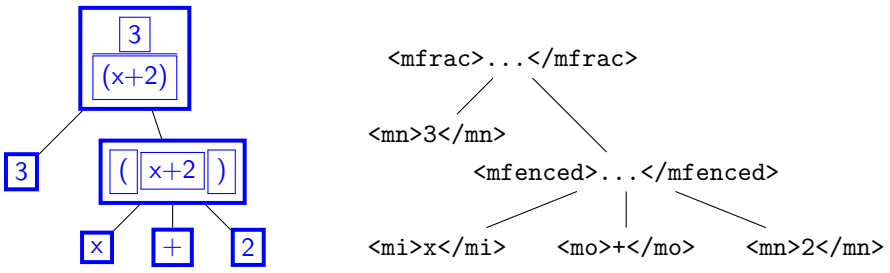
- ▷ Mathematical Expressions are build up as expression trees
  - ▷ of layout schemata in Presentation-MathML
  - ▷ of functional subexpressions in Content-MathML
- ▷ Example: $\frac{3}{x+2}$

```
<mfrac>
  <mn>3</mn>
  <mfenced>
     <mi>x</mi>
     <mo>+</mo>
     <mn>2</mn>
   </mfenced>
</mfrac>
```

```
<apply>
  <divide/>
  <cn>3</cn>
  <apply>
    <plus/>
    <ci>x</ci>
    <cn>2</cn>
  </apply>
</apply>
```


Layout Schemata and the MathML Box model

- ▷ Presentation MathML represents the visual appearance of a formula in a tree of layout primitives
- ▷ **Example 24.2.1 (Presentation MathML for $3/(x+2)$)**



P-MathML Token Elements

- ▷ Tokens Elements directly contain character data(the only way to include it)
  Attributes: `fontweight`, `fontfamily` and `fontstyle`, `color`...
- ▷ Identifiers: `<mi>... </mi>` (∼ variables, italicized)
- ▷ Numbers: `<mn>... </mn>` (numbers)
- ▷ Operators: `<mo>... </mo>` (constants, functions, upright)
- ▷ Operator display is often ideosyncratic (Operator Dictionaries for defaults)
  - ▷ Examples: spacing, *-scripts in sums and limits, stretchy integrals,...
  - ▷ Attributes: `lspace`, `rspace`, `stretchy`, and `movablelimits`.
  - ▷ Operators include delimiter characters like
    - ▷ parentheses (which stretch),
    - ▷ punctuation (which has uneven spacing around it) and
    - ▷ accents (which also stretch).

General Layout Schemata

- ▷ horizontal row: `<mrow>child1 ... </mrow>` (alignment and grouping)
- ▷ fraction: `<mfrac>numerator denominator </mfrac>`
  Attribute: `linethickness` (set to 0 for binomial coefficients)
- ▷ Radicals: `<msqrt>child1 ... </msqrt>` and
  `<mroot>base index</mroot>`
- ▷ grouping with parenthesis: `<mfenced>child ... </mfenced>`
  Attributes: `open="("` and `close="]"` to specify parentheses
- ▷ grouping and style: `<mstyle>child ... </mstyle>` (pre-set attributes)

Example: $x^2 + 4x + 4 = 0$

just presentation	some structure

`<mrow>`
`  <msup>`
`  <mi>x</mi>`
`  <mn>2</mn>`
`  </msup>`
`  <mo>+</mo>`
`  <mn>4</mn>`
`  <mi>x</mi>`
`  <mo>+</mo>`
`  <mn>4</mn>`
`  <mo>=</mo>`
`  <mn>0</mn>`
`</mrow>`
	`<mrow>`
	`  <mrow>`
	`  <msup>`
	`  <mi>x</mi>`
	`  <mn>2</mn>`
	`  </msup>`
	`  <mo>+</mo>`
	`  <mrow>`
	`  <mn>4</mn>`
	`  <mi>x</mi>`
	`  </mrow>`
	`  <mo>+</mo>`
	`  <mn>4</mn>`
	`  </mrow>`
	`  <mo>=</mo>`
	`  <mn>0</mn>`
	`</mrow>`

Example: Grouping Arguments by `mfenced`

$f(x+y)$	$f(x+y)$

`<mrow>`
`  <mi>f</mi>`
`  <mfenced>`
`  <mrow>`
`  <mi>x</mi>`
`  <mo>+</mo>`
`  <mi>y</mi>`
`  </mrow>`
`  </mfenced>`
`</mrow>`
	`<mrow>`
	`  <mi>f</mi>`
	`  <mfenced>`
	`  <mstyle color='#ff0000'>`
	`  <mrow>`
	`  <mi>x</mi>`
	`  <mo>+</mo>`
	`  <mi>y</mi>`
	`  </mrow>`
	`  </mstyle>`
	`  </mfenced>`
	`</mrow>`

Example: `<mfrac>` and `mroot`

```
<mroot>
  <mrow>
    <mn>1</mn>
    <mo>-</mo>
    <mfrac>
      <mi>x</mi>
      <mn>2</mn>
    </mfrac>
  </mrow>
  <mn>3</mn>
</mroot>
```

$\sqrt[3]{1-\frac{x}{2}}$

Example: The quadratic formula $x = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$

```
<mrow>
  <mi>x</mi>
  <mo>=</mo>
  <mfrac>
    <mrow>
      <mrow><mo>-</mo><mi>b</mi></mrow>
      <mo>&plusmn;</mo>
      <msqrt>
        <mrow>
          <msup><mi>b</mi><mn>2</mn></msup>
          <mo>-</mo>
          <mrow><mn>4</mn><mi>a</mi><mi>c</mi></mrow>
        </mrow>
      </msqrt>
    </mrow>
    <mrow><mn>2</mn><mo>&InvisibleTimes;</mo><mi>a</mi></mrow>
  </mfrac>
</mrow>
```


Script Schemata

- ▷ Indices: G^1, H_5, $R^i_j \ldots$
  - ▷ Super: `<msup>base script </msup>`
  - ▷ Subs: `<msub>base script </msub>`
  - ▷ Both: `<msubsup>base superscript subscript</msub>`(vertical alignment!)
- ▷ Bars and Arrows: $\overline{X}$, $\underbrace{Y}$, $\overrightarrow{Z}$,...
  - ▷ Under: `<munder>base script</munder>`
  - ▷ Over: `<mover>base script</mover>`
  - ▷ Both: `<munderover>base underscript overscript </munderover>`

▷ Tensor-like: <mmultiscripts>base sub1 sup1 ... [<mprescripts/>psub1 psup1 ...] </mmultiscript

msub + msup vs. msubsup

msub + msup	msubsup
<msup> <msub> <mi>x</mi> <mn>1</mn> </msub> <mi>α</mi> </msup>	<msubsup> <mi>x</mi> <mn>1</mn> <mi>α</mi> </msubsup>
$x_1{}^{\alpha}$	x_1^{α}

Example: Movable Limits on Sums

```
<mrow>
  <mstyle displaystyle='true'>
    <munderover>
      <mo>&sum;</mo>
      <mrow><mi>i</mi><mo>=</mo><mn>1</mn></mrow>
      <mi>&infty;</mi>
    </munderover>
    <msup><mi>x</mi><mi>i</mi></msup>
  </mstyle>
  <mo>+</mo>
  <mstyle displaystyle='false'>
    <munderover>
      <mo>&sum;</mo>
      <mrow><mi>i</mi><mo>=</mo><mn>1</mn></mrow>
      <mi>&infty;</mi>
    </munderover>
    <msup><mi>x</mi><mi>i</mi></msup>
  </mstyle>
</mrow>
```

$$\sum_{i=1}^{\infty} x^i + \sum_{i=1}^{\infty} x^i$$

24.3 Content MathML

Content Mathml: Expression Trees in Prefix Notation

▷ Prefix Notation saves parentheses (so does postfix, BTW)

$(x-y)/2$	$x-(y/2)$

`<apply>`
`  <divide/>`
`  <apply>`
`  <minus/>`
`  <ci>x</ci>`
`  <ci>y</ci>`
`  </apply>`
`  <cn>2</cn>`
`</apply>`
	`<apply>`
	`  <minus/>`
	`  <ci>x</ci>`
	`  <apply>`
	`  <divide/>`
	`  <ci>y</ci>`
	`  <cn>2</cn>`
	`  </apply>`
	`</apply>`

Function Application: `<apply>function arg1 ... argn </apply>`

▷▷ Operators and Functions: ~ 100 empty elements `<sin/>`, `<plus/>`, `<eq/>`, `<compose/>`,...

▷ Token elements: `ci`, `cn` (identifiers and numbers)

▷ Extra Operators: `<csymbol cd="...">...</csymbol>`

Containers (aka Constructors)

▷ sets: `<set><elt1><elt2>... </set>` or `<set><bvar>...</bvar><condition>...</condition></set>`

▷ intervals: `<interval><pt1><pt2></interval>`
Attribute: `closure` (one of `open`, `closed`, `open-closed`, `closed-open`)

▷ vectors: `<vector><elt1><elt2>... </vector>`

▷ matrix rows: `<matrixrow><elt1><elt2>... </matrixrow>`

▷ matrices: `<matrix><row1><row2>... </matrix>`

Examples of Content Math

Expression	Markup

`<apply>`
`  <plus/>`
`  <apply><sin/><ci>x</ci></apply>`
`  <cn>9</cn>`
`</apply>`
	$\sin(x) + 9$

`<apply><eq/><ci>x</ci><cn>1</cn></apply>`	$x = 1$

`<apply><sum/>`
`  <bvar><ci>n</ci></bvar>`
`  <lowlimit><cn>0</cn></lowlimit>`
`  <uplimit><ci>&infty;</ci></uplimit>`
`  <apply><power/><ci>x</ci><ci>n</ci></apply>`
`</apply>`
	$\sum_0^\infty x^n$

`<apply><diff/>`
` <bvar><ci>x</ci><degree><cn>3</cn></degree></bvar>`
` <apply><ci>f</ci><ci>x</ci></apply>`
`</apply>`
	$\frac{d^3}{dx^3} f(x)$

`<set>`
` <bvar><ci>x</ci></bvar>`
` <bvar><ci>y</ci></bvar>`
` <condition>`
`  <apply><and/>`
`  <apply><lt/><ci>0</ci><ci>x</ci><ci>1</ci></apply>`
`  <apply><leq/><ci>3</ci><ci>y</ci><ci>10</ci></apply>`
`  </apply>`
` </condition>`
`</set>`
	$\left\{ x, y \;\middle\vert\; \begin{array}{l} 0 < x < 1, \\ 3 \leq y \leq 10 \end{array} \right\}$

Expression	Markup
<apply><eq/> <set> <bvar><ci>x</ci></bvar> <condition> <apply><geq/><ci>x</ci><cn>0</cn></apply> </condition> </set> <interval closure='closed-open'> <cn>0</cn> <cn>&infty;</cn> </interval> </apply>	$\{x\|x \geq 0\} = [0, \infty)$
<apply> <eq/> <apply><times/> <vector><cn>1</cn><cn>2</cn></vector> <matrix> <matrixrow><cn>0</cn><cn>1</cn></matrixrow> <matrixrow><cn>1</cn><cn>0</cn></matrixrow> </matrix> </apply> <apply> <transpose/> <vector><cn>2</cn><cn>1</cn></vector> </apply> </apply>	$(1,2) \times \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = (2,1)^t$

Chapter 25

Converting the arXiv

The arXMLiv Project: arXiv to semantic XML

- ▷ Idea: Develop a large corpus of knowledge in OMDoc/PhysML
  - ▷ to get around the chicken-and-egg problem of MKM
  - ▷ corpus-linguistic methods for semantics recovery (linguists interested)
- ▷ **Definition 25.0.1 (The Cornell Preprint arXiv)** (http://www.arxiv.org)
  Open access to ca. 850K e-prints in Physics, Mathematics, Computer Science and Quantitative Biology.
- ▷ **Definition 25.0.2 (The arXMLiv Project)** (http://arxmliv.kwarc.info)
  - ▷ use Bruce Miller's LaTeXML to transform to XHTML+MathML
  - ▷ extend to LaTeXML daemon (RESTful web service)(http://latexml.mathweb.org)
  - ▷ we have an automated, distributed build system (ca. Q2CPU-years)
  - ▷ create ca. 12K LaTeXML binding files (8 Jacobs students help)
  - ▷ use MathWebSearch to index XML version (realistic search corpus)
- ▷ More semantic information will enable more added-value services, e.g.
  - ▷ filter hits by model assumptions(expanding, stationary, or contracting universe)
  - ▷ use linguistic techniques to add the necessary semantics

Why reimplement the TeX parser?

- ▷ Problem: The TeX parser can change the tokenizer while at runtime(\catcode)
- ▷ **Example 25.0.3 (Obfuscated TeX)** David Carlisle posted the following, when someone claimed that word counting is simple in TeX/LaTeX

```
\let~\catcode~‘76~‘A13~‘F1~‘j00~‘P2jdefA71F~‘7113jdefPALLF
PA’’FwPA;;FPAZZFLaLPA//71F71iPAHHFLPAzzFenPASSFthP;A$$RevP
```

```
A@@FfPARR717273F737271P;ADDFRgniPAWW71FPATTFvePA**FstRsamP
AGGFRruoPAqq71.72.F717271PAYY7172F727171PA??Fi*LmPA&&71jfi
Fjfi71PAVVFjbigskipRPWGAUU71727374 75,76Fjpar71727375Djifx
:76jelse&U76jfiPLAKK7172F7117271PAXX71FVLnOSeL71SLRyadR@oL
RrhC?yLRurtKFeLPFovPgaTLtReRomL;PABB71 72,73:Fjif.73.jelse
B73:jfiXF71PU71 72,73:PWs;AMM71F71diPAJJFRdriPAQQFRsreLPAI
I71Fo71dPA!!FRgiePBt'el@ lTLqdrYmu.Q.,Ke;vz vzLqpip.Q.,tz;
;Lql.IrsZ.eap,qn.i. i.eLlMaesLdRcna,;!;h htLqm.MRasZ.ilk,%
s$;z zLqs'.ansZ.Ymi,/sx ;LYegseZRyal,@i;@ TLRlogdLrDsW,@;G
LcYlaDLbJsW,SWXJW ree @rzchLhzsW,;WERcesInW qt.'oL.Rtrul;e
doTsW,Wk;Rri@stW aHAHHFndZPpqar.tridgeLinZpe.LtYer.W,:jbye
```

When formatted by TeX, this leads to the full lyrics of "The twelve days of christmas". When formattet by LaTeXML, it gives

```
<song>
  <verse>
    <line>On the first day of Christmas my true love gave to me</line>
    <line>a partridge in a pear tree.</line>
  </verse>
  <verse>
    <line>On the second day of Christmas my true love gave to me</line>
    <line>two turtle doves</line>
    <line>and a partridge in a pear tree.</line>
  </verse>
  <verse>
    <line>On the third day of Christmas my true love gave to me</line>
    <line>three french hens</line>
    <line>two turtle doves</line>
    <line>and a partridge in a pear tree.</line>
  </verse>
  <verse>
    <line>On the fourth day of Christmas my true love gave to me</line>
    <line>four calling birds</line>
    <line>three french hens</line>
    <line>two turtle doves</line>
    <line>and a partridge in a pear tree.</line>
  </verse>
  ...
```

But the real reason is: that we can take advantage of the semantics in the LaTeX.

▷▷ LaTeXML does not need to expand macros, we can tell it about XML equivalents.

▷ **Example 25.0.4 (Recovering the Semantics of Proofs)**
Add the following magic incantation to `amsthm.sty.ltxml`(LaTeXML binding)

```
DefEnvironment('{proof}',"<xhtml:div class='proof'>#body</xhtml:div>");
```

The arXMLiv approach: Try to cover most packages and classes in the arXiv (Jacobs undergrads' intro to research)

▷ Future Plans for arXMLiv

- ▷ State: LaTeX-to-XHTML+MathML Format Conversion works(65% success)
- ▷ Over the summer: Bump up success rate to 75%, daily downloads, web site, instrumentation,...
- ▷ Soon: Integrate user-level quality control(integrate JS feedback into html)
- ▷ starting Fall: Extend post-processing by linguistic methods for semantic analysis
  - ▷ build semantics blackboard/database for linguistic information(rdf triples)
  - ▷ extend build system for arbitrary XML2BB processes
  - ▷ invite the linguists over (they leave semantics results in BB)
  - ▷ harvest the semantics BB to get OMDoc representations

Current and Possible Applications

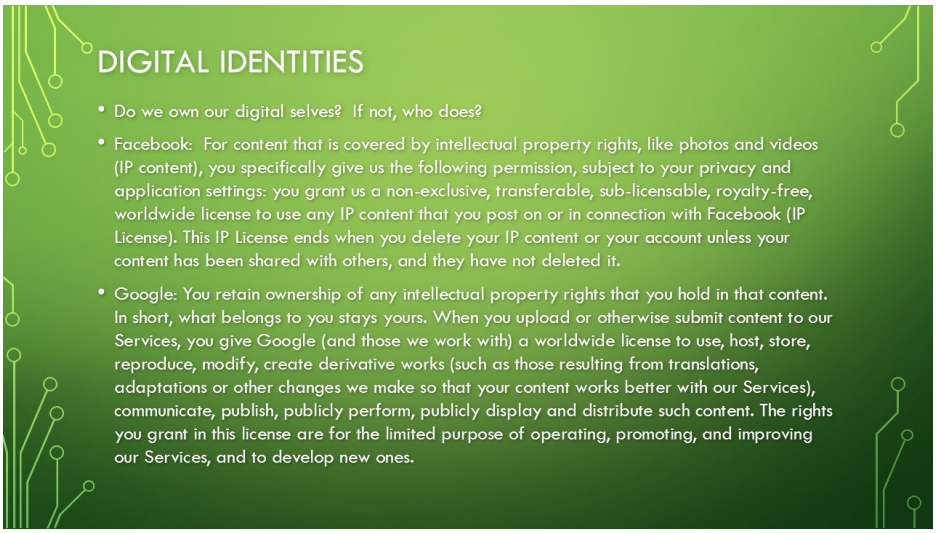
- ▷ the arxmliv build system `http://arxmliv.kwarc.info`
- ▷ the transformation web service `http://tex2xml.kwarc.info`
- ▷ LaTeXML daemon to avoid `perl` and LaTeX startup times (Deyan Ginev)
  - ▷ keep LaTeXML alive as a daemon that can process multiple files/fragments (patch memory leaks)
  - ▷ a LaTeXML client just passes files/fragments along (10/s to 100/s)
- ▷ embedding/editing LaTeX in web pages `http://tex2xml.kwarc.info/test`
- ▷ a MathML version of the arXiv allows vision-impared readers to understand the texts
- ▷ generalization search (need to know sentence structure for detecting universal variables)
- ▷ semantic search by academic discipline or theory assumption(need discourse structure)
- ▷ development of scientific vocabularies(over the past 18 years; drink from the source)

Chapter 26

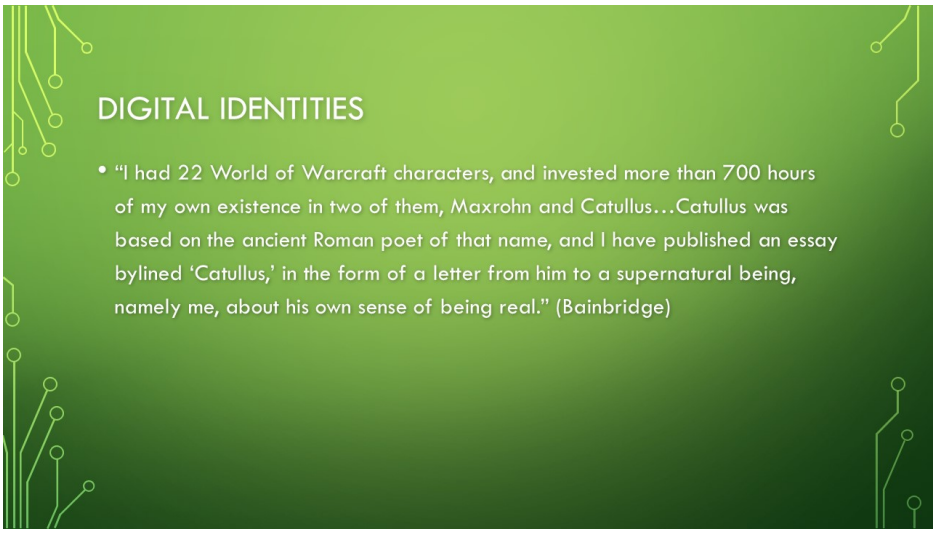
Virtual Immortality

Slide 316

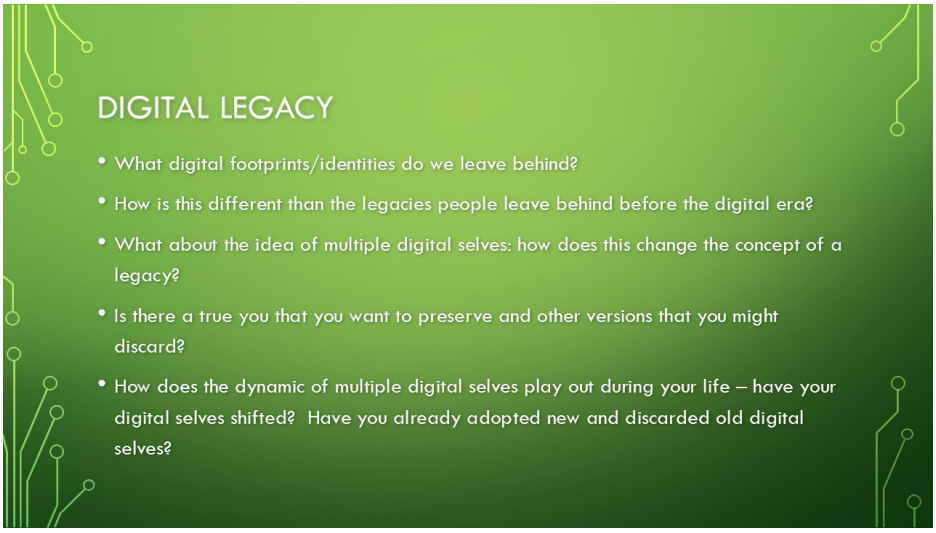
Slide 317



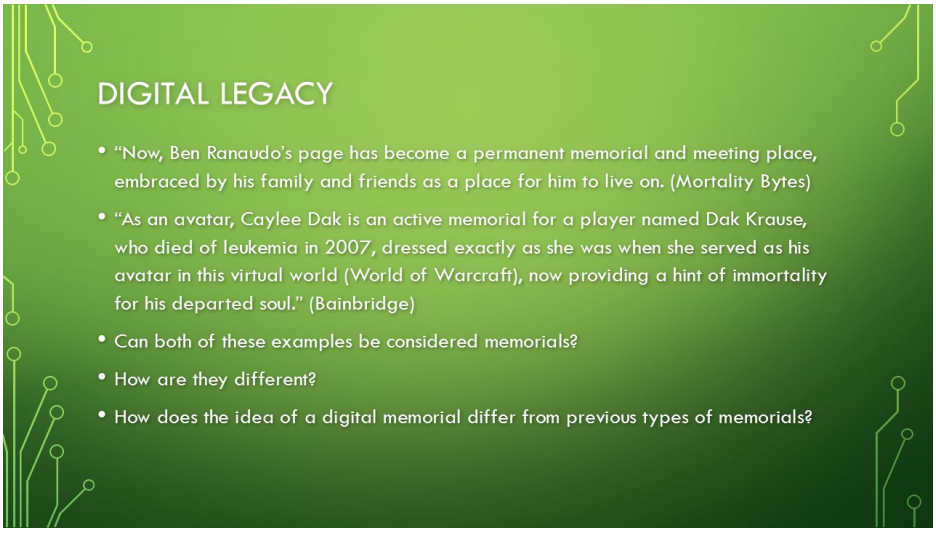
Slide 318



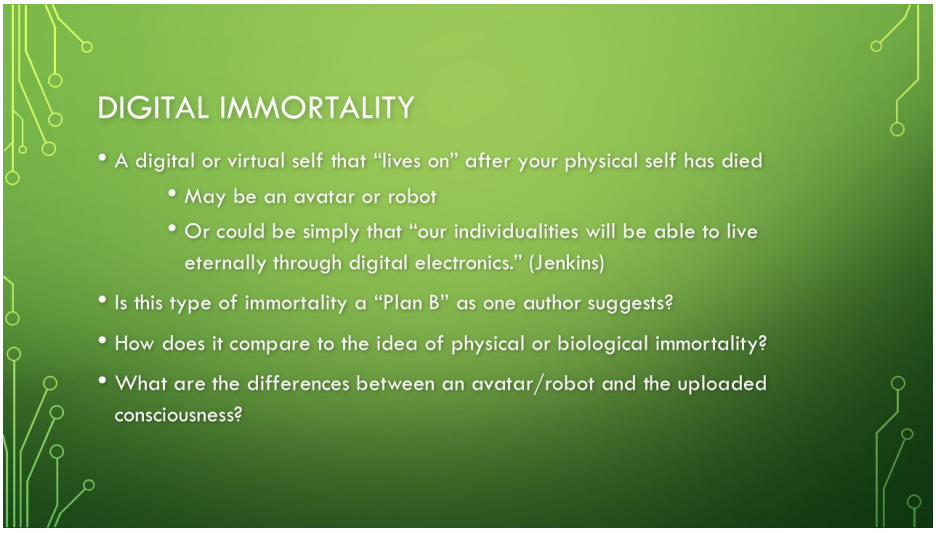
Slide 319



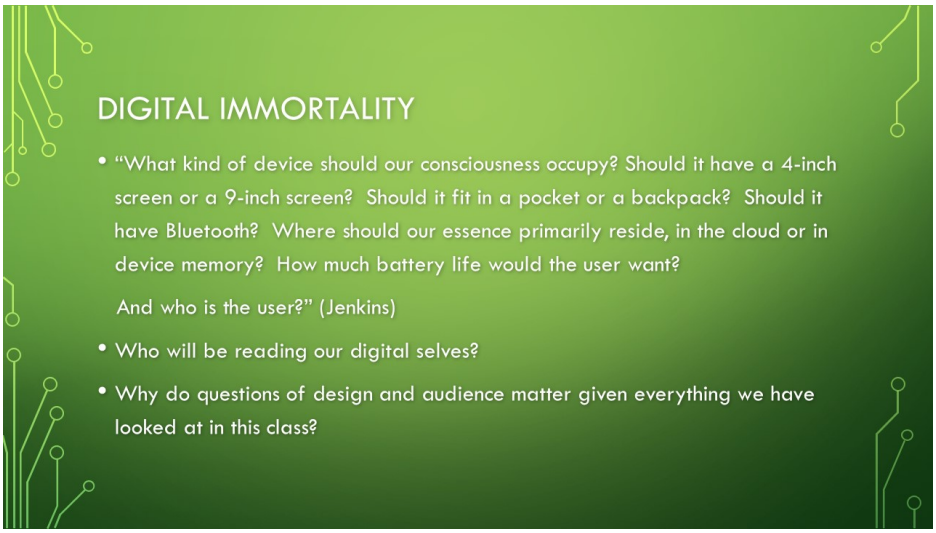
Slide 320



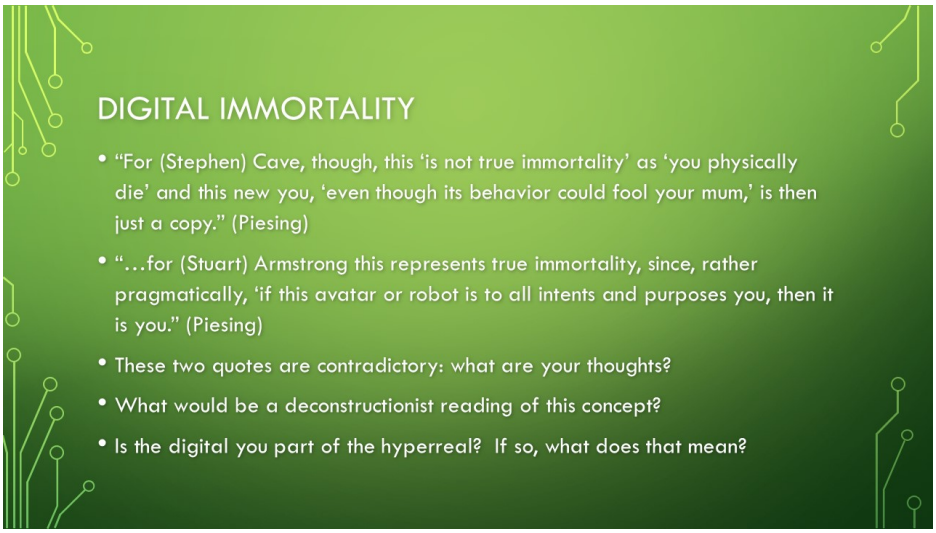
Slide 321



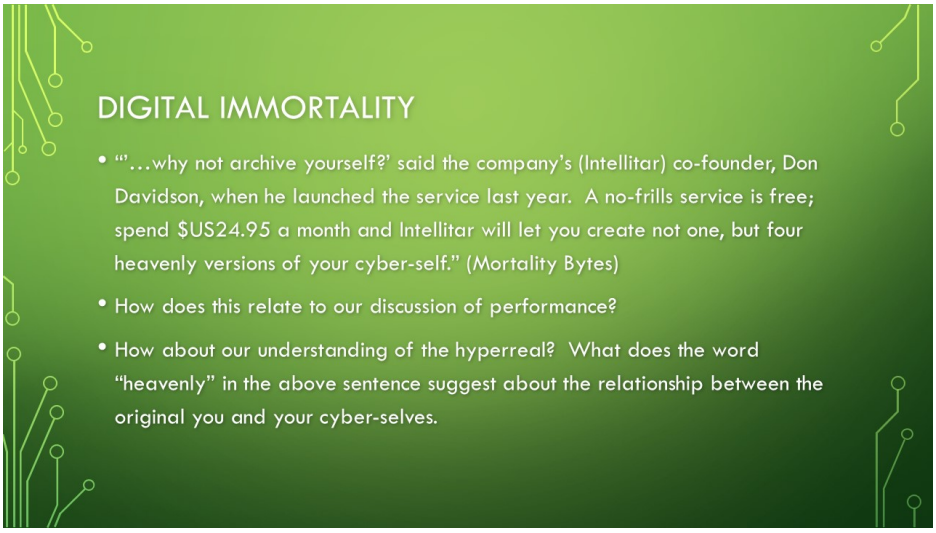
Slide 322



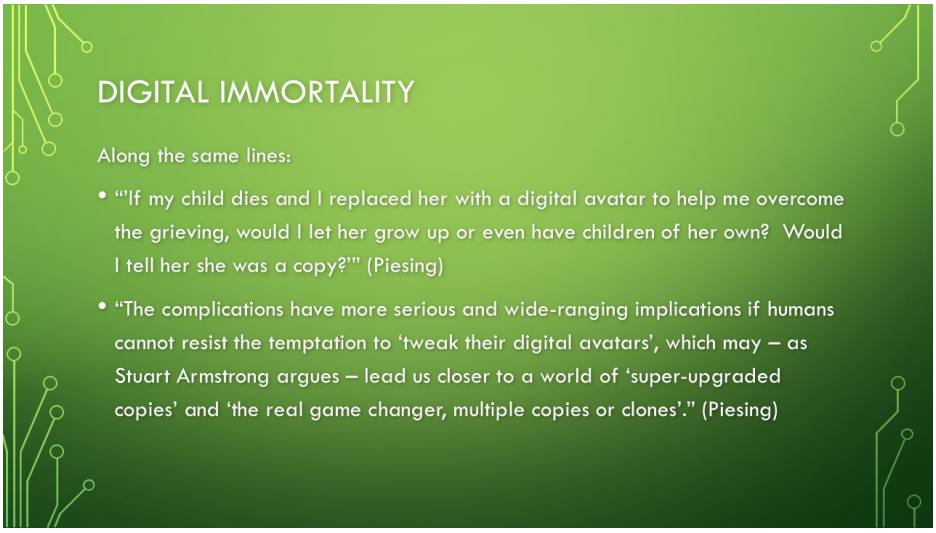
Slide 323



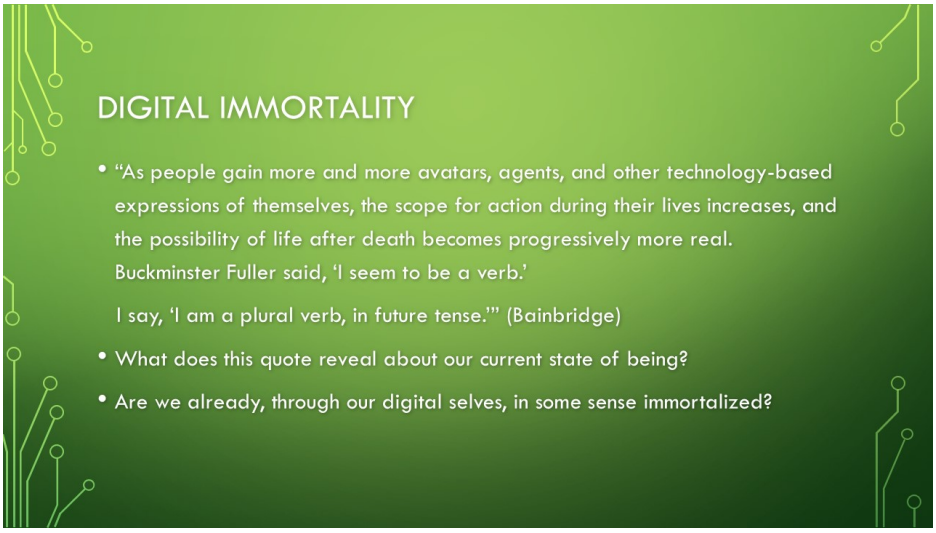
Slide 324



Slide 325



Slide 326



Slide 327

Chapter 27

Active Documents

27.1 Planetary: A Social Semantic eScience System

The PLANETARY System

▷ The PLANETARY system is a Web 3.0 system for semantically annotated document collections in Science, Technology, Engineering and Mathematics (STEM).

▷ Web 3.0 stands for extension of the Social Web with Semantic Web/Linked Open Data technologies.

▷ documents published in the PLANETARY system become flexible, adaptive interfaces to a content commons of domain objects, context, and their relations.

▷ PLANETARY is based on the Active Documents Paradigm (see next)

▷ **Example 27.1.1 (Example installments)**

- ▷ `arxivdemo.mathweb.org` (presentation/structural Level: arXiv)
- ▷ `panta.kwarc.info` (semantic level: PantaRhei course system)
- ▷ `logicatlas.omdoc.org` (fully formal level: Logic Representations)
- ▷ `planetbox.kwarc.info` (Technology Sandbox)

▷ The PLANETARY system is finalist in the Elsevier Executable Papers Challenge.

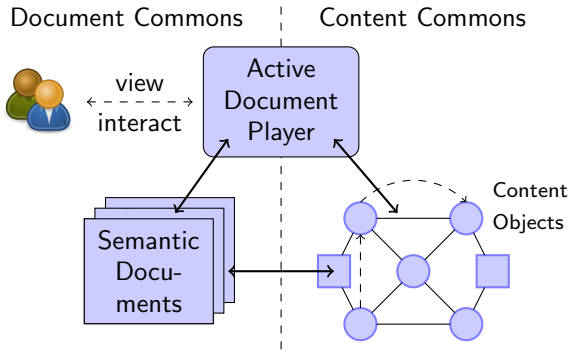
The Active Documents Paradigm

▷ **Definition 27.1.2** The active documents paradigm (ADP) consists of

- ▷ *semantically annotated documents* together with
- ▷ background ontologies (which we call the content commons),

▷ *semantic services* that use this information

▷ a document player application that embeds services to make documents executable.

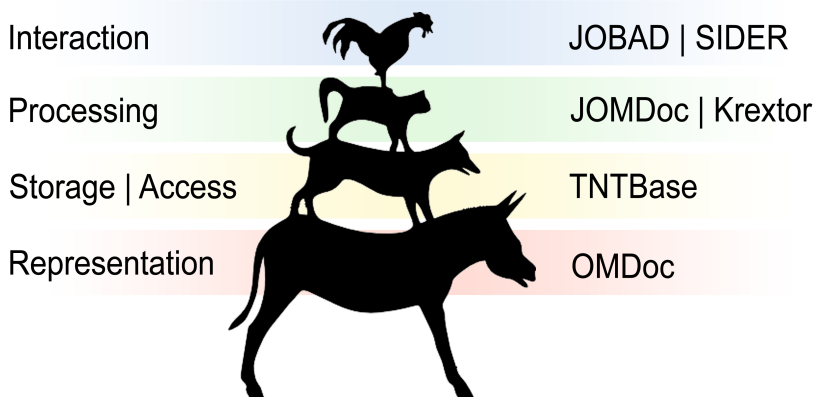


▷ **Example 27.1.3** Services can be program (fragment) execution, computation, visualization, navigation, information aggregation and information retrieval

27.2 Realizing Planetary

Realizing PLANETARY: The KWARC stack

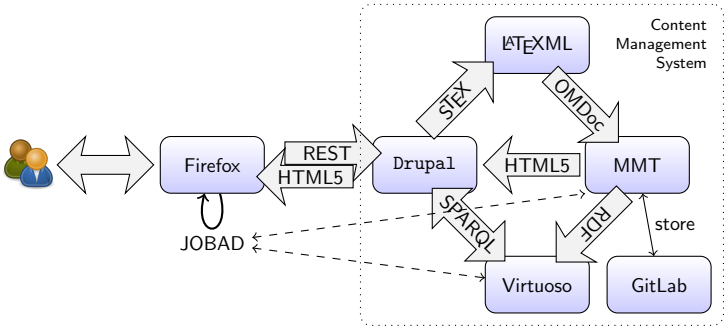
We have already developed the necessary tools/systems over the last decade



PLANETARY is the ideal test bed to integrate them.

Assembling PLANETARY: System Architecture

▷ PLANETARY functionality can be achieved by integrating existing components.

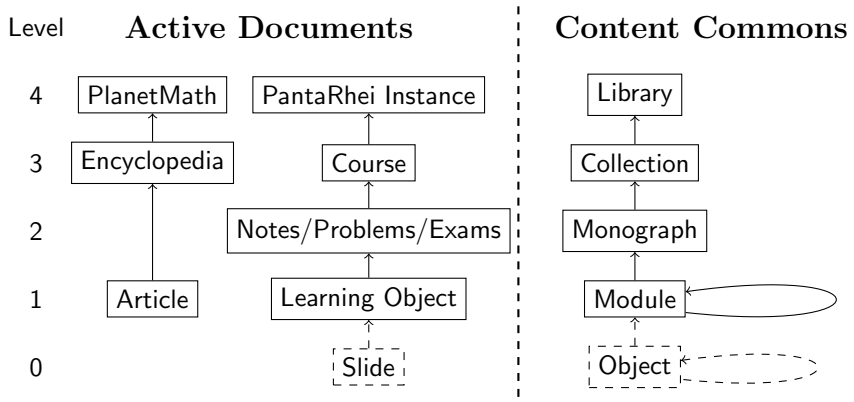


- Drupal for discussions, user management, caching,
- TNTBase for versioned XML storage, OMDoc presentation
- JOBAD integrates semantic services into documents
- Virtuoso is a triple store for semantic relations
- LaTeXML transforms LaTeX/sTeX to XHTML+MathML+RDFa

27.2.1 Organization of Content/Narrative Structure

Layers of Documents/Content

- Content and narrative structures come at different conceptual layers



- Different layers support different functionality

The lowest level consists of atomic "modules"[1], i.e. content objects that correspond to small (active) documents dedicated to a single topic. For a course management system these might be learning objects (either as single modules or module trees), for an encyclopedia these would be

[1]The level of objects below modules consists of individual statements (e.g. definitions, model assumptions, theorems, and proofs), semantic phrase-level markup, and formulae. Even though it carries much of the semantic relations, it does not play a great role for the document-level phenomena we want to discuss here in this paper.

the individual articles introducing a topic. Note that technically, we allow modules to contain (denoted by the arrows) other modules, so that larger discourse structures could be formed. For example, sections can be realized as modules referencing other modules of subsections, etc.

The next level up is the level of "monographs", written works on a single subject that have a complete, self-contained narrative structure, usually by a single author or group of authors who feel responsible for the whole monograph. As a content object, a monograph is usually built up from modules, e.g. as a "module tree" that corresponds to sectioning structure of traditional books, but often also includes front and backmatter such as a preface, acknowledgements (both special kinds of modules), table of contents, lists of tables and figures, an index and references (generated from content annotations). Course notes in the PantaRhei system are typical examples, while other documents at the monograph level are articles in a journal, or books in a certain topical section of a library.

Multiple monographs can be combined into collections, adding special modules for editorial comments, etc. Concrete collections in the document realm are encyclopedias, academic journals, conference proceedings, or courses in a course management system.

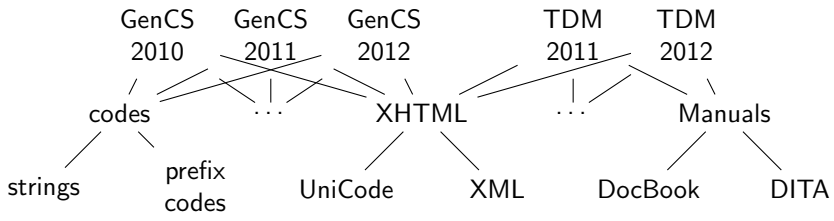
Finally, the library level collects and grants access to collections, concrete, modern-day examples are digital libraries, installed course management systems, etc. In practice, a library provides a base URI that establishes the web existence of the particular installation. In the Semantic Web world, the library is the authority that makes its resources addressable by URLs.

Monographs as Module Graphs foster Reuse

▷ Idea: Modules can be reused in more than one monograph

▷ Note: Similar to, but more general (nesting) than DITA concepts and DITA maps. (but no conditional processing (yet))

▷ **Example 27.2.1** For instance a module on HTML/XML in the courses "General Computer Science" and "Text and Digital Media".



Courses given in different years share most of their content (but not all)

▷ Observation: These graphs can get quite large: Our corpus has 3300 nodes with 130 roots.

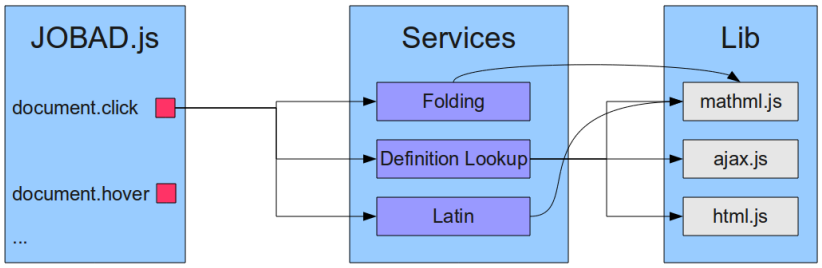
JOBAD: Embedding Semantic Services into Web Docs

▷ JavaScript API for (J)OMDoc Based Active Documents

▷ runs inside client browser (FireFox currently)

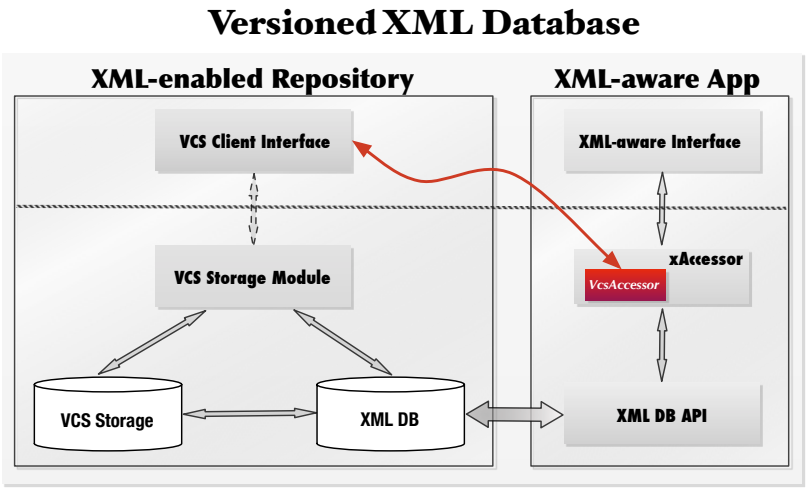
▷ provides client-only or server-based features (extensible framework) based on semantic annotations in XHTML+MathML+RDFa documents

▷ Project home page: `https://jomdoc.omdoc.org/wiki/JOBAD`



TNTBase: Versioned Storage for XML

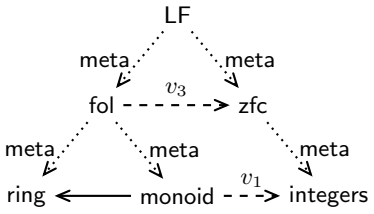
▷ The TNTBase system is a versioned storage system for XML documents. It combines the functionality and interfaces of Subversion with those of an XML database.



OMDoc in a Nutshell (three levels of modeling) [Koh06]

Formula level: OpenMath/C-MathML
▷ Objects as logical formulae
▷ symbol meaning by reference to theory level
	`<apply>`
	` <csymbol cd="ring">plus</c.>`
	` <csymbol cd="ring">zero</c.>`
	` <ci>N</ci>`
	`</apply>`

Statement level:
▷ Definition, Theorem, Proof, Example
▷ semantics via explicit forms and refs.
▷ parallel formal & natural language
	`<defn for="plus" type="rec">`
	` <CMP>rec. eq. for plus</CMP>`
	` <FMP>`$X + 0 = X$`</FMP>`
	` <FMP>`$X + s(Y) = s(X + Y)$`</FMP>`
	`</defn>`

Module level: Theory Graph [RK13] ▷ inheritance via symbol-mapping ▷ views by proof-obligations ▷ logics as meta-theories (logic atlas) ▷ meta-logics as oracles for type/eq	

LaTeXML: Converting TeX/LaTeX Documents to XML

▷ **Definition 27.2.2** LaTeXML converts LaTeX documents to XHTML+MathML

- ▷ re-implement the TeX parser in `perl`.(do not expand semantic macros)
- ▷ needs LaTeXML bindings for all LaTeX packages and classes(specify the XML for the emitter)

Case Study: Converting the arXiv into XHTML+MathML(70% coverage of 550 k documents)

▷ sTeX, a Semantic Variant of TeX/LaTeX

▷ Problem: Need content markup formats for semantic services, but Mathematicians write LaTeX

▷ Idea: Enable the author to make structure explicit and disambiguate meanings

- ▷ use the TeX macro mechanism for this (well established)
- ▷ the author knows the semantics best (at least she understands)
- ▷ the burden is is alleviated by manageability savings(MKM on TeX/LaTeX)

▷ **Definition 27.2.3 (sTeX Approach)** Semantic pre-loading of TeX/LaTeX documents.

- ▷ Introduce semantic macros: e.g. `\union{a,b,c}` $\rightsquigarrow a \cup b \cup c$

▷ Mark up discourse structure: (largely invisible)
e.g. \begin{sproof}[id=Wiles,for=Fermat]...\end{sproof}
▷ Generate PDF and OMDoc from that (via LaTeXML [Mil])

http://trac.kwarc.info/sTeX/

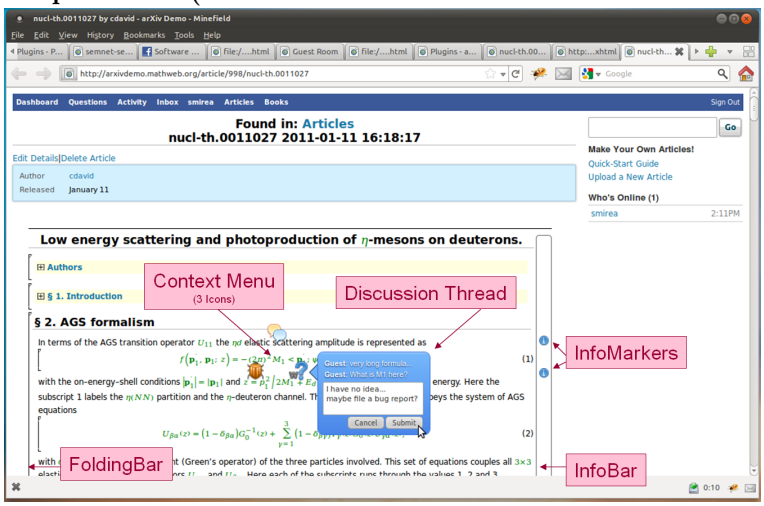
27.3 Levels of Service in Planetary

The importance of the presentation structure level is that PLANETARY can turn legacy documents into active documents by transforming them into XHTML+MathML+SVG-encoded documents with semantic annotations in RDFa. We have transformed over half a million articles from the Cornell ePrint arXiv to XHTML+MathML with LaTeXML, preserving properties like document and formula structures and embedded them into an instance of the PLANETARY system.

PLANETARY at the Presentation/Structural Level

▷ PLANETARY can make use objects and relations at various levels,

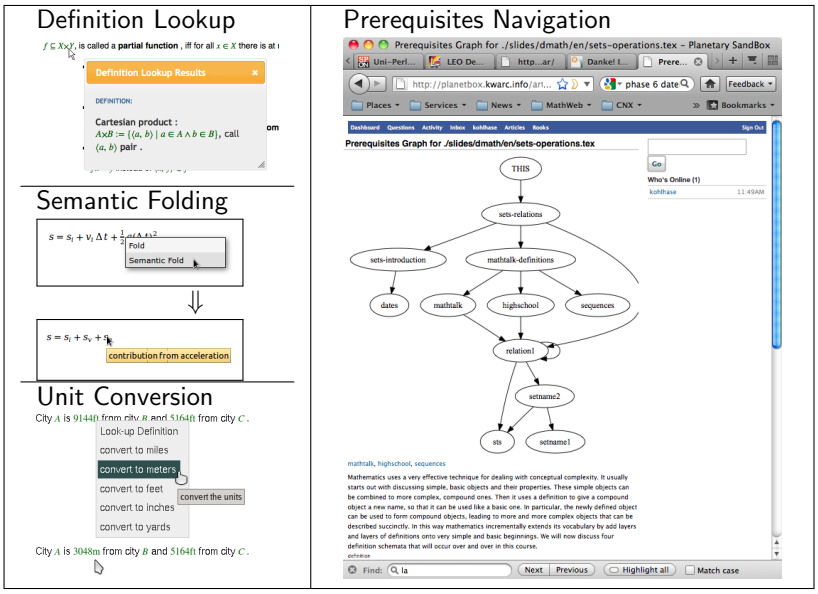
▷ **Example 27.3.1 (arXivdemo: Document Structure and Presentational Math)**



The document structure can then be exploited for a folding bar service (see on the left in Figure **??**) and for localizing discussions about document content to document structures and subformulae – e.g. for questions/answers, or reviewers' comments. In the situation in Figure **??** we have clicked on formula (1), which pops up the icon menu with three options: reporting errors in the content (bug icon), asking/answering a question (question mark icon), and accessing the discussion threads of this element (balloons icon). Here, a click on the question mark icon allowed us to pose a question and hope for an anser by other users in the forum. Figure **??** also shows the PLANETARY infobar with information markers on the right, which indicate the availability and state of the discussion threads pertaining to information objects in the line they are horizontally aligned with. Clicking them will highlight all items that have discussions. Localized discussions have proven a very valuable tool for community-based validation of papers, especially if they are coupled

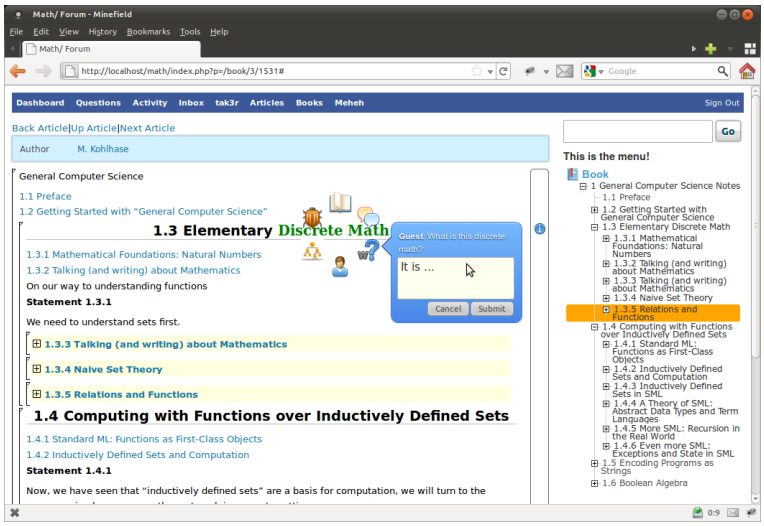
with a discussion subscription/trackback system for readers and personal notification system for authors.

User Services at the Semantic Level in PLANETARY



PantaRhei: Semantic Course Knowledge Exploration

▷ PantaRhei is a semantic course knowledge exploration system based on the PLANETARY system.

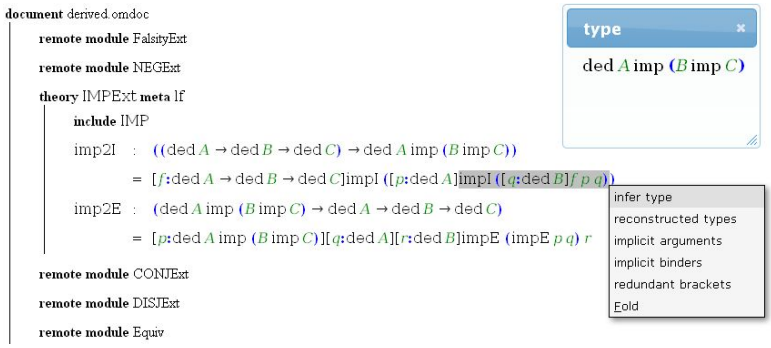


User Services at the Formal Level in PLANETARY

▷ Formal Level: e.g. specification and verification in the LATIN Logic Atlas

- ▷ flexible elision of brackets
- ▷ argument reconstruction via external TWELF system
- ▷ verification by the HETS system

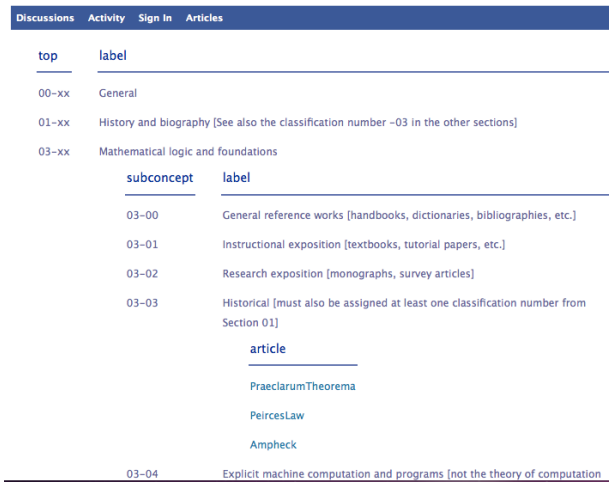
▷ **Example 27.3.2** Adapting formal representations user preferences



Accessing Encyclopedias via Ontologies

▷ Idea: add classification metadata to articles, harvest as RDF into triple store,
compute access methods via SPARQL queries and SKOS ontology.

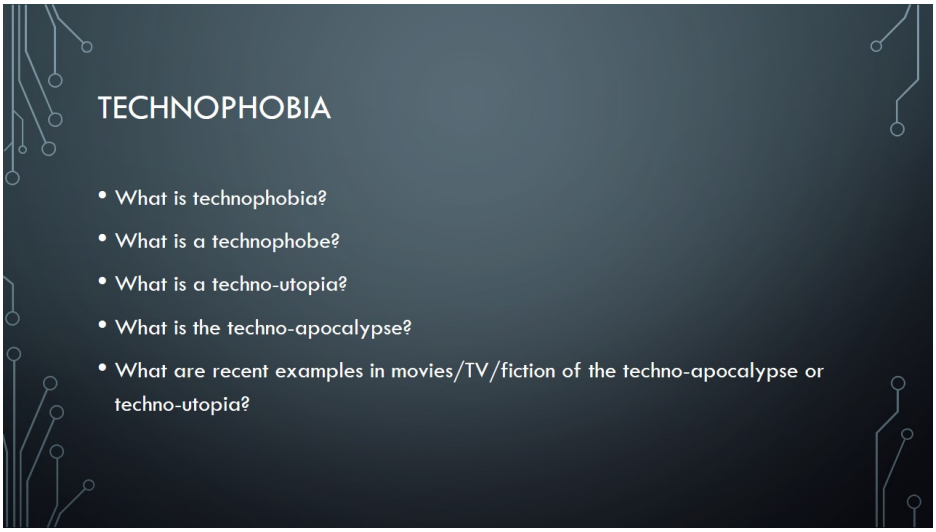
▷ **Example 27.3.3 (MSC View in Planet Math)** use the Math Subject Classification



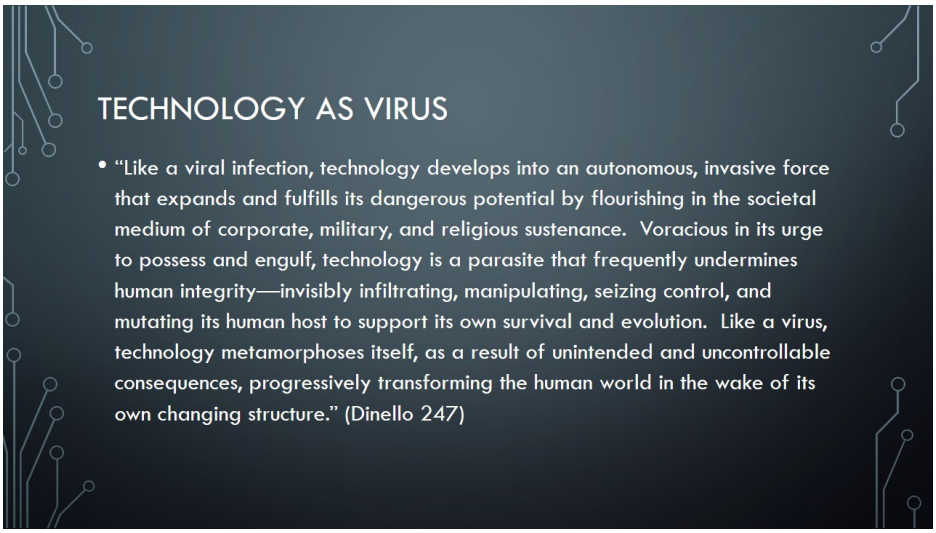
Chapter 28

Zombie Apocalypse

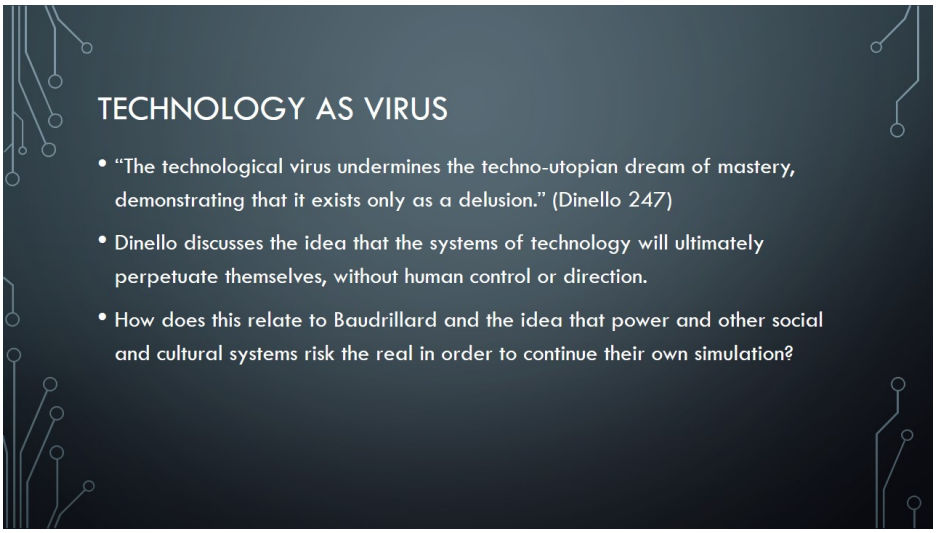
Slide 344



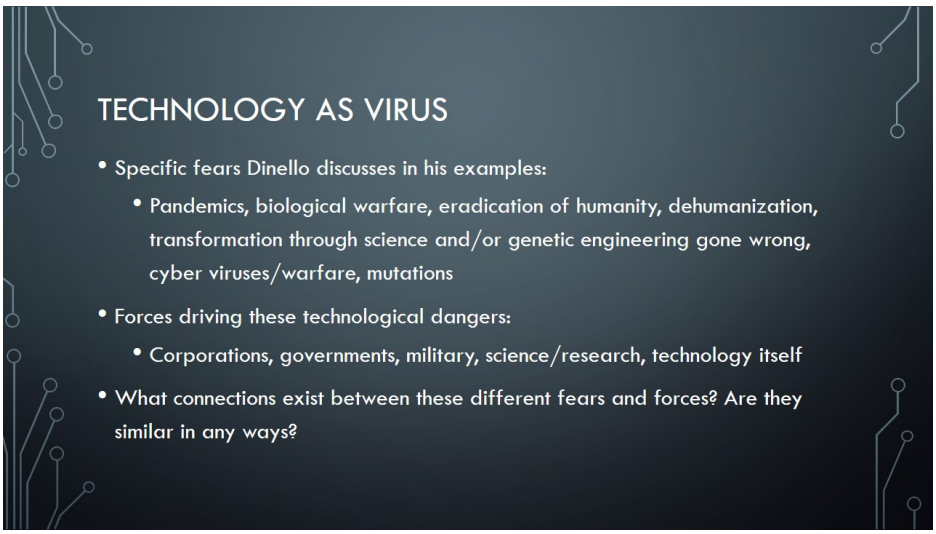
Slide 345



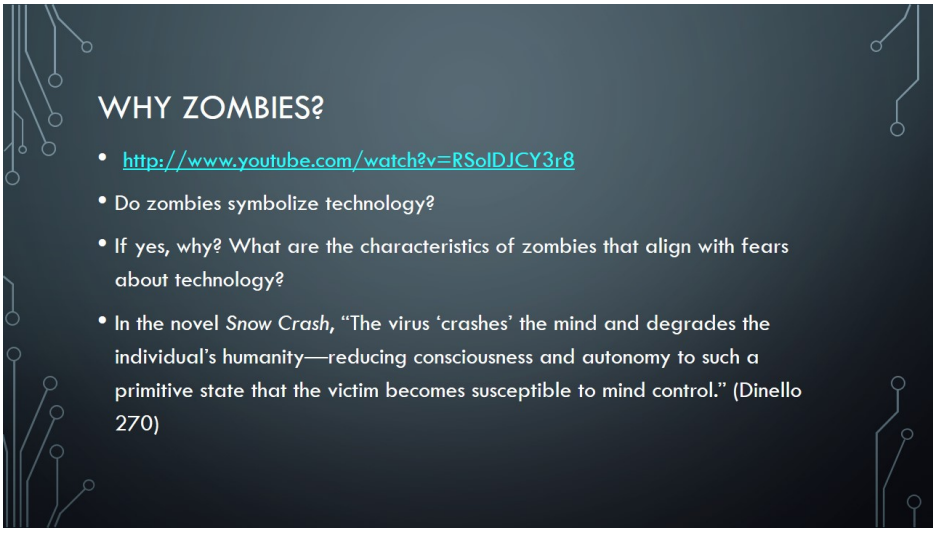
Slide 346



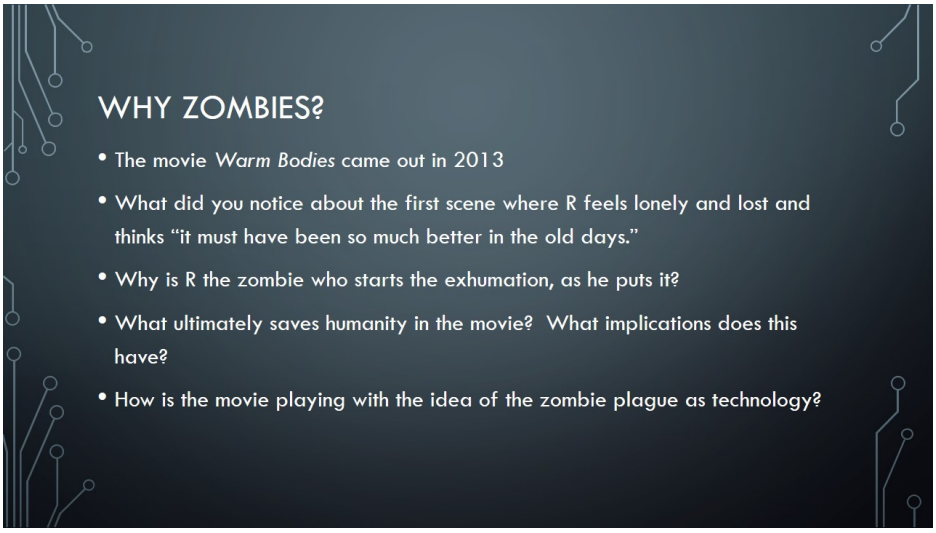
Slide 347



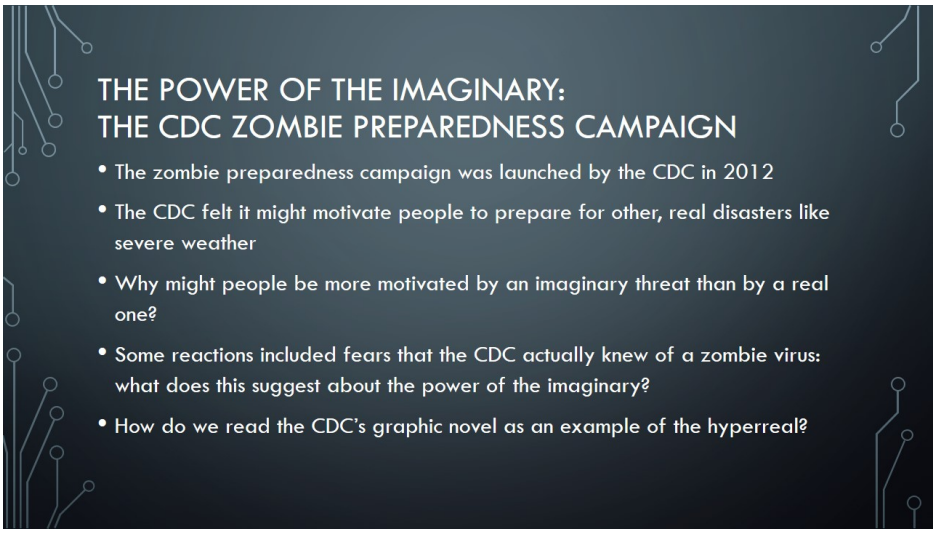
Slide 348



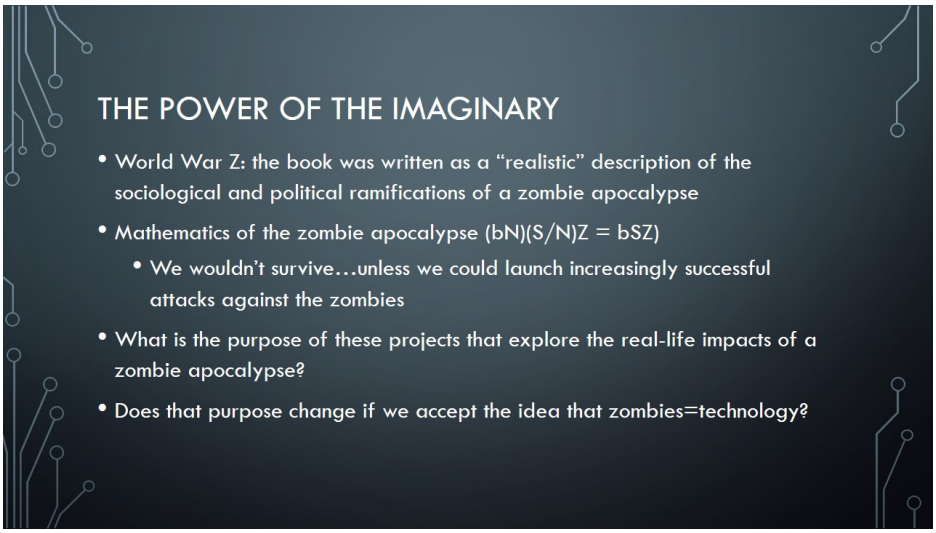
Slide 349



Slide 350



Slide 351



Slide 352

Slide 353

Bibliography

[BCHL09] Bert Bos, Tantek Celik, Ian Hickson, and Høakon Wium Lie. Cascading style sheets level 2 revision 1 (CSS 2.1) specification. W3C Candidate Recommendation, World Wide Web Consortium (W3C), 2009.

[BLFM05] Tim Berners-Lee, Roy T. Fielding, and Larry Masinter. Uniform resource identifier (URI): Generic syntax. RFC 3986, Internet Engineering Task Force (IETF), 2005.

[CQ69] Allan M. Collins and M. Ross Quillian. Retrieval time from semantic memory. *Journal of verbal learning and verbal behavior*, 8(2):240–247, 1969.

[ECM09] ECMAScript language specification, December 2009. 5$^{\text{th}}$ Edition.

[FGM+99] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext transfer protocol – HTTP/1.1. RFC 2616, Internet Engineering Task Force (IETF), 1999.

[HASB13] Ivan Herman, Ben Adida, Manu Sporny, and Mark Birbeck. RDFa 1.1 primer – second edition. W3C Working Goup Note, World Wide Web Consortium (W3C), 2013.

[HL11] Martin Hilbert and Priscila López. The world's technological capacity to store, communicate, and compute information. *Science*, 331, feb 2011.

[Kay08] Michael Kay. Saxonica: XSLT and XQuery processing. `http://www.saxonica.com`, 2008.

[KC04] Graham Klyne and Jeremy J. Carroll. Resource Description Framework (RDF): Concepts and abstract syntax. W3C recommendation, World Wide Web Consortium (W3C), 2004.

[Koh06] Michael Kohlhase. OMDoc – *An open markup format for mathematical documents [Version 1.2]*. Number 4180 in LNAI. Springer Verlag, August 2006.

[Koh08] Michael Kohlhase. Using LaTeX as a semantic markup format. *Mathematics in Computer Science*, 2(2):279–304, 2008.

[Koh13] Michael Kohlhase. sTeX: Semantic markup in TeX/LaTeX. Technical report, Comprehensive TeX Archive Network (CTAN), 2013.

[Mil] Bruce Miller. `LaTeXML`: A LaTeX to XML converter. Web Manual at `http://dlmf.nist.gov/LaTeXML/`. seen September2011.

[OWL09] OWL Working Group. OWL 2 web ontology language: Document overview. W3C recommendation, World Wide Web Consortium (W3C), October 2009.

[PRR97] G. Probst, St. Raub, and Kai Romhardt. *Wissen managen*. Gabler Verlag, 4 (2003) edition, 1997.

[RHJ98] Dave Raggett, Arnaud Le Hors, and Ian Jacobs. HTML 4.0 Specification. W3C Recommendation REC-html40, World Wide Web Consortium (W3C), April 1998.

[RK13] Florian Rabe and Michael Kohlhase. A scalable module system. *Information & Computation*, 0(230):1–54, 2013.

[Vei] Daniel Veillard. The xslt c library for gnome; the xsltproc tool. System Home page at `http://xmlsoft.org/XSLT/xsltproc2.html`.

[Vol11] Victor Volkman. Classic parsing with flex and bison. `http://www.codeguru.com/csharp/.net/net_general/patterns/article.php/c12805__2/Classic-Parsing-with-Flex-and-Bison.htm`, 2011. visited Feb 2011.

[Wik11] Wikipedia. Epub — wikipedia, the free encyclopedia, 2011. [Online; accessed 15-March-2011].

[XAM] apache friends - xampp. `http://www.apachefriends.org/en/xampp.html`.

[XML] Extensible Markup Language (XML) 1.0 (Fourth Edition). Web site at `http://www.w3.org/TR/REC-xml/`.

Index